PATENT COOPERATION TREATY

From the INTERNATIONAL BUREAU

PCT

NOTIFICATION OF ELECTION

(PCT Rule 61.2)

To:

Assistant Commissioner for Patents
United States Patent and Trademark Office
Box PCT
Washington, D.C.20231
ETATS-UNIS D'AMERIQUE

in its capacity as elected Office

Date of mailing (day/month/year) 12 October 2000 (12.10.00)	
International application No. PCT/US00/01634	**Applicant's or agent's file reference** 3COM 2257-3
International filing date (day/month/year) 24 January 2000 (24.01.00)	**Priority date** (day/month/year) 25 January 1999 (25.01.99)
Applicant LUO, Wenjun et al	

1. The designated Office is hereby notified of its election made:

[X] in the demand filed with the International Preliminary Examining Authority on:

25 August 2000 (25.08.00)

[ ] in a notice effecting later election filed with the International Bureau on:

2. The election [X] was

[ ] was not

made before the expiration of 19 months from the priority date or, where Rule 32 applies, within the time limit under Rule 32.2(b).

The International Bureau of WIPO
34, chemin des Colombettes
1211 Geneva 20, Switzerland

Facsimile No.: (41-22) 740.14.35

Authorized officer

S. Mafla

Telephone No.: (41-22) 338.83.38

PATENT COOPERATION TREATY

PCT

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

(PCT Article 36 and Rule 70)

Applicant's or agent's file reference 3COM 2257-3	**FOR FURTHER ACTION**	See Notification of Transmittal of International Preliminary Examination Report (Form PCT/IPEA/416)
International application No. PCT/US00/01634	International filing date *(day/month/year)* 04 JANUARY 2000	Priority date *(day/month/year)* 25 JANUARY 1999
International Patent Classification (IPC) or national classification and IPC IPC(7): G06F 13/00 and US Cl.: 709/219, 225, 328		
Applicant 3COM CORPORATION		

1. This international preliminary examination report has been prepared by this International Preliminary Examining Authority and is transmitted to the applicant according to Article 36.

2. This REPORT consists of a total of 3 sheets.

   ☐ This report is also accompanied by ANNEXES, i.e., sheets of the description, claims and/or drawings which have been amended and are the basis for this report and/or sheets containing rectifications made before this Authority. (see Rule 70.16 and Section 607 of the Administrative Instructions under the PCT).

   These annexes consist of a total of 0 sheets.

3. This report contains indications relating to the following items:

   I ☒ Basis of the report

   II ☐ Priority

   III ☐ Non-establishment of report with regard to novelty, inventive step or industrial applicability

   IV ☐ Lack of unity of invention

   V ☒ Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

   VI ☐ Certain documents cited

   VII ☐ Certain defects in the international application

   VIII ☐ Certain observations on the international application

Date of submission of the demand	Date of completion of this report
25 AUGUST 2000	15 MARCH 2001
Name and mailing address of the IPEA/US Commissioner of Patents and Trademarks Box PCT Washington, D.C. 20231 Facsimile No. (703) 305-3230	Authorized officer VIET VU James R. Matthews Telephone No. (703) 305-9600

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.
PCT/US00/01634

I. Basis of the report

1. With regard to the elements of the international application:*

[ ] the international application as originally filed

[X] the description:
pages 1-53 , as originally filed
pages 54 , filed with the demand
pages NONE , filed with the letter of _____

[X] the claims:
pages 56-62 , as originally filed
pages NONE , as amended (together with any statement) under Article 19
pages NONE , filed with the demand
pages NONE , filed with the letter of _____

[X] the drawings:
pages 1-12 , as originally filed
pages NONE , filed with the demand
pages NONE , filed with the letter of _____

[X] the sequence listing part of the description: NONE , as originally filed
pages NONE , filed with the demand
pages NONE , filed with the letter of _____

2. With regard to the language, all the elements marked above were available or furnished to this Authority in the language in which the international application was filed, unless otherwise indicated under this item.
These elements were available or furnished to this Authority in the following language _____ which is:

[ ] the language of a translation furnished for the purposes of international search (under Rule 23.1(b)).

[ ] the language of publication of the international application (under Rule 48.3(b)).

[ ] the language of the translation furnished for the purposes of international preliminary examination (under Rules 55.2 and/or 55.3).

3. With regard to any nucleotide and/or amino acid sequence disclosed in the international application, the international

[ ] contained in the international application in printed form.

[ ] filed together with the international application in computer readable form.

[ ] furnished subsequently to this Authority in written form.

[ ] furnished subsequently to this Authority in computer readable form.

[ ] The statement that the subsequently furnished written sequence listing does not go beyond the disclosure in the international application as filed has been furnished.

[ ] The statement that the information recorded in computer readable form is identical to the writen sequence listing has been furnished.

4. [X] The amendments have resulted in the cancellation of:

[X] the description, pages NONE

[X] the claims, Nos. NONE

[X] the drawings, sheets/~~fig~~ NONE

5. [ ] This report has been drawn as if (some of) the amendments had not been made, since they have been considered to go beyond the disclosure as filed, as indicated in the Supplemental Box (Rule 70.2(c)).**

** Replacement sheets which have been furnished to the receiving Office in response to an invitation under Article 14 are referred to in this report as "originally filed" and are not annexed to this report since they do not contain amendments (Rules 70.16 and 70.17).*

*** Any replacement sheet containing such amendments must be referred to under item 1 and annexed to this report.*

INTERNATIONAL PRELIMINARY EXAMINATION REPORT

International application No.
PCT/US00/01634

V. Reasoned statement under Article 35(2) with regard to novelty, inventive step or industrial applicability; citations and explanations supporting such statement

1. statement

Novelty (N)	Claims	1-44	YES
	Claims	NONE	NO
Inventive Step (IS)	Claims	1-7, 13, 18	YES
	Claims	8-12, 14-17, 19-44	NO
Industrial Applicability (IA)	Claims	1-44	YES
	Claims	NONE	NO

2. citations and explanations (Rule 70.7)

1. Claims 8-12, 14-17, 19-44 lack an inventive step under PCT Article 33(3) as being obvious over Ezekiel, U.S. pat. no. 5,790,977 in view of Kuzma, U.S. pat. no. 5,832,506.

Per claims 8-12, 14-17 and 19-25, Ezekiel discloses a method for controlling an application executable on a remote network device comprising:

a) establishing a communication link between a computing platform and the network device (see col 4, lines 12-21),

b) transferring a control program to the computing platform via the network, the control program including a user interface for controlling the network device (col 4, lines 43-51),

c) transmitting commands to the network device,

d) transferring the commands to the application (see col 5, lines 31-47).

Ezekiel does not teach accessing a directory of services. The use of a proxy server to provide directory of services in the network in well-known in the art as disclosed in Kuzma (see Kuzma's col 1, lines 29-52).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to utilize such directory server in Ezekiel because it would have enabled users to locate various services available in the network.

Per claims 26-34, it is also well-known in the art that different types of services can be made available via the network.

Per claims 35-44, it would have been further obvious to one skilled in the art to implement Ezekiel's invention on any computing platforms and/or any conventional communication links (see Ezekiel's col 2, lines 5-8).

2. Claims 1-7, 13 and 18 meet the criteria set out in PCT Article 33(2)-(4), because the prior art of record does not teach or fairly suggest using a palm-sized computer to control an application program executed on a remote network device via a wireless communication link.

---------- NEW CITATIONS ----------

NONE

PCT

WORLD INTELLECTUAL PROPERTY ORGANIZATION
International Bureau

INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(51) International Patent Classification 7 : G06F 13/00

A1

(11) International Publication Number: WO 00/43891

(43) International Publication Date: 27 July 2000 (27.07.00)

(21) International Application Number: PCT/US00/01634

(22) International Filing Date: 24 January 2000 (24.01.00)

(30) Priority Data:
09/237,609 25 January 1999 (25.01.99) US

(63) Related by Continuation (CON) or Continuation-in-Part (CIP) to Earlier Application
US 09/237,609 (CIP)
Filed on 25 January 1999 (25.01.99)

(71) Applicant *(for all designated States except US)*: 3COM CORPORATION [US/US]; 5400 Bayfront Plaza, Santa Clara, CA 95052 (US).

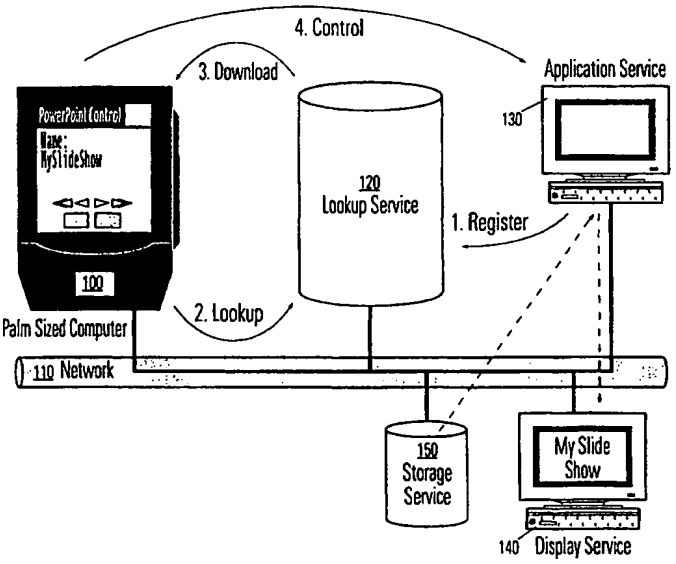
(72) Inventors; and
(75) Inventors/Applicants *(for US only)*: LUO, Wenjun [CN/US]; 40664 Witherspoon Terrace, Fremont, CA 94538 (US). LUSHER, Elaine, P. [US/US]; 456 Montori Court, Pleasanton, CA 94566 (US).

(74) Agent: HAYNES, Mark, A.; Haynes & Beffel LLP, P.O. Box 366, Half Moon Bay, CA 94019 (US).

(81) Designated States: CA, GB, JP, US, European patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE).

Published
With international search report.

(54) Title: SYSTEM AND METHOD TO CONTROL NETWORK DEVICES ON A PALM SIZED COMPUTER



(57) Abstract

Controlling network services using palm sized computers is described. A program on the palm sized computer (100) is used to access a registry (120) of network services that may be available. The registry (120) includes descriptions for various services. Each description includes at least a reference to program code that can be downloaded to the palm sized computer. Executing this program causes the palm sized computer to issue commands directly to the specific network services needed. In some cases, these network services include application services for running desktop applications (140) that the palm sized computer could not execute.

FOR THE PURPOSES OF INFORMATION ONLY

Codes used to identify States party to the PCT on the front pages of pamphlets publishing international applications under the PCT.

AL	Albania	**ES**	Spain	**LS**	Lesotho	**SI**	Slovenia
AM	Armenia	**FI**	Finland	**LT**	Lithuania	**SK**	Slovakia
AT	Austria	**FR**	France	**LU**	Luxembourg	**SN**	Senegal
AU	Australia	**GA**	Gabon	**LV**	Latvia	**SZ**	Swaziland
AZ	Azerbaijan	**GB**	United Kingdom	**MC**	Monaco	**TD**	Chad
BA	Bosnia and Herzegovina	**GE**	Georgia	**MD**	Republic of Moldova	**TG**	Togo
BB	Barbados	**GH**	Ghana	**MG**	Madagascar	**TJ**	Tajikistan
BE	Belgium	**GN**	Guinea	**MK**	The former Yugoslav Republic of Macedonia	**TM**	Turkmenistan
BF	Burkina Faso	**GR**	Greece			**TR**	Turkey
BG	Bulgaria	**HU**	Hungary	**ML**	Mali	**TT**	Trinidad and Tobago
BJ	Benin	**IE**	Ireland	**MN**	Mongolia	**UA**	Ukraine
BR	Brazil	**IL**	Israel	**MR**	Mauritania	**UG**	Uganda
BY	Belarus	**IS**	Iceland	**MW**	Malawi	**US**	United States of America
CA	Canada	**IT**	Italy	**MX**	Mexico	**UZ**	Uzbekistan
CF	Central African Republic	**JP**	Japan	**NE**	Niger	**VN**	Viet Nam
CG	Congo	**KE**	Kenya	**NL**	Netherlands	**YU**	Yugoslavia
CH	Switzerland	**KG**	Kyrgyzstan	**NO**	Norway	**ZW**	Zimbabwe
CI	Côte d'Ivoire	**KP**	Democratic People's Republic of Korea	**NZ**	New Zealand		
CM	Cameroon			**PL**	Poland		
CN	China	**KR**	Republic of Korea	**PT**	Portugal		
CU	Cuba	**KZ**	Kazakstan	**RO**	Romania		
CZ	Czech Republic	**LC**	Saint Lucia	**RU**	Russian Federation		
DE	Germany	**LI**	Liechtenstein	**SD**	Sudan		
DK	Denmark	**LK**	Sri Lanka	**SE**	Sweden		
EE	Estonia	**LR**	Liberia	**SG**	Singapore		

SYSTEM AND METHOD TO CONTROL NETWORK DEVICES ON A PALM SIZED COMPUTER

LIMITED COPYRIGHT WAIVER

BACKGROUND OF THE INVENTION

Field of the Invention

This invention relates to the field of networked computer. In particular, the invention relates to a system and method to control network devices using a palm sized, or otherwise reduced functionality, computer.

Description of the Related Art

Palm sized computers, also referred to as Personal Digital Assistants (PDAs), are portable devices which perform an array of personal management tasks such as calendar management and address book storage. The adoption of palm sized computers has been rapid. Some palm sized computers are able to interface with conventional computing devices, such as PCs, on an as-needed basis. For example, palm sized computers such as 3Com's Palm Platform™ computers can upload personal appointments to a PC-based calendar.

Palm sized computers generally have the following characteristics. Relative to desktop and laptop computers, palm sized computers have limited processing, display and input capabilities. As a result of these limitations, palm sized computers do not run the same applications as desktop or laptop computers. Other limitations of palm sized computer include limited batter life and lower bandwidth communications with other devices.

One big advantage of palm sized computers is their portability. Therefore, it is desirable to be able to access desktop functionality from palm sized computers.

SUMMARY OF THE INVENTION

Controlling network services using palm sized computers is described. A program on the palm sized computer is used to access a registry of network services that may be available. The registry includes descriptions for various services. Each description includes at least a reference to program code that can be downloaded to the palm sized computer. Executing this program causes the palm sized computer to issue commands directly to one or more of the specific network services needed. In some cases, these network services include application services for running desktop applications that the palm sized computer could not execute.

In some embodiments, the device executing the network services and the palm sized computer are executing middleware application for communicating with the registry. In some embodiments, this middleware includes Jini technology from Sun Microsystems. Additionally, the programs downloaded can include Java program code.

BRIEF DESCRIPTION OF THE FIGURES

Figure 1 illustrates a system having a palm sized computer controlling operations of various network devices.

Figure 2 illustrates a software architecture for use in the system of Figure 1.

Figure 3 illustrates an example architecture for controlling a PowerPoint presentation.

Figure 4 illustrates a detailed software architecture for the example of Figure 3.

Figure 5 illustrates a class hierarchy that can be used in the example of Figure 3.

Figure 6 illustrates an example class hierarchy for network services that can be provided in the system of Figure 1.

Figure 7 illustrates an example class hierarchy for application services that can be provided in the system of Figure 1.

Figure 8 is an architecture diagram for one embodiment of the present invention.

Figures 9-16 are representative Graphic User Interface constructs for one example of the present invention for the control of a slide presentation application.

DETAILED DESCRIPTION

A. System Overview

A palm sized computer, or other thin platforms having small sizes and displays with display areas typically smaller that 4 inches by 6 inches, can serve as a *network* portal to usher in a new generation of mobile computing. Palm sized computers are the ideal next generation computing device in that they are inherently mobile and have the lightweight form factor necessary for mobile computing. The fundamental obstacle to palm sized computers as the next generation computing device can be removed by viewing the network as an extension of the palm sized computer's resources. Functions can be downloaded into the device as needed, and overlaid after they have been used. This allows the palm sized computer to adapt to a changing environment (as the mobile user's location changes) and to access exactly the set of services it needs. These services are stored on the network and can be used at will. Many of these services may never reside on the device and are more suitable for execution by a conventional computer. However, they are *accessible* and can be *controlled* via a lightweight computing device, such as a palm sized computer.

B. Building Blocks of Network-enabled Palm Sized Computers

The building blocks for lightweight mobile computing include a palm sized computer, a compute/memory/storage-intensive device(s), and a network.

Middleware allows palm sized computers to discover network-based computing resources. Once discovered, this middleware provides a mechanism for the palm sized computer to use these resources. This middleware typically includes a directory of resources (or services), a protocol for storing and retrieving from the directory, and mechanisms to transfer software from the directory to a palm sized computer.

Control applications reside on a lightweight computing device, such as a palm sized computer, but manipulate computing services on the network. Control applications assume the use of middleware, such as Sun Microsystems Jini, for access to network-

based resources. (Note in some embodiments, the control application is distributed between the palm sized computer and a control proxy server.)

Example Registry and Control Infrastructure: Jini

Jini™ is a technology developed by Sun Microsystems which addresses the problem of computing and network complexity. It eases the burden of accessing services by providing seamless access and automatic registration of network resources. Jini accomplishes this by adopting a model in which devices announce their presence and capabilities to the network, and access the same type of information in order to locate services they wish to use.

The Jini approach relies on Java and a Jini "registry" (i.e. database of services) as the underlying infrastructure. Each device is expected to run a Java Virtual Machine (JVM), or rely on a Jini proxy which runs a JVM on the device's behalf. Key to Java is the idea that software (as well as data) can be dynamically downloaded to a device. The Java model assumes a distributed, network-centric model in which the behavior of a device can be dynamically altered to accommodate changing conditions.

Jini eases network connectivity problems. Jini acts as middleware to access network resources, as it lets devices locate services and download software for those services. Other middleware could be substituted for Jini if it provides discovery and software download for network-based services.

C. Definitions

A control device is as a device which runs Jini (or some other discovery and software download technology) and is capable of accessing a wide range of network-centric services, including services which are incapable of running on the control device. The control device is the access device for a wide class of computing services otherwise inaccessible to devices with restricted computing power. 3Com's palm sized computer, the Palm Computing platform, is an example of such a control device. Windows CE compatible devices may also be used.

A control application is as an application that relies on resources located off of the control device (e.g. on a network), but uses a control device to initiate and control the application. The resources are accessed and controlled, but not resident, on the control device. Examples of such compute/memory-intensive services include PowerPoint slide presentations

and speech recognition systems. These services can be both invoked and controlled via a control device.

Network-based services (rather than devices) can be used by any control device. Services offer a discrete task to the control device. This contrasts with a traditional model in which devices, such as a PC, house the entirety of resources a control device might utilize. Services are fine-grained and represent a distributed set of capabilities residing on a network. Services may or may not be co-located with other services on the same physical device. Services are offered (via a network and a Directory of Services, such as the Jini Lookup) in as small a unit as is justifiable given the application tasks users will typically want to accomplish. Fine-grained services can be used by a variety of consumers who need precisely that function. The model that emerges is a network-based model that offers a wide range of narrowly-defined computing services of interest to a wide range of applications. Services will be physically distributed across devices but accessible via a central repository (e.g. database) of services. This model of distributed computing is inherently scalable in that the set of services offered via the network grows seamlessly as devices "plug in" and announce their capabilities.

D. An Example of the Control Application

To help illustrate the use of control device and a control application, a PowerPoint slide presentation control described herein. (Figure 1 illustrates a generalized view of such an example.) In this example, a palm sized computer 100 accesses network 110 resources (via a Directory of Services, such as the lookup service 120) to locate the elements it requires to remotely control a presentation located on the network. The palm sized computer 100 uses three services to function as a control device to display a PowerPoint presentation:

- an application service 130 (to run PowerPoint)
- a persistent storage service 150 (to store the presentation)
- a display service 140 (to display the presentation images)

Each of these services will have registered with the lookup service 100. A service in this example, is tied to different hardware devices. However, this is not required. Storage, processing, and/or display could be supported by one or more computer systems.

None of these services are resident on the palm sized computer 100. Once the palm sized computer 100 has located the necessary services, it downloads the code required to

control those services (using the lookup and download protocols). Middleware, such as Sun's Java/Jini technology, is used to move the code.

The palm sized computer 100 is then capable of directly controlling the services it requires.

In this example, all the devices can communicate on the network 110, however, they need not all be connected to the network in the same way.

Functionally, the devices play the following roles:

- palm sized computer 100: the remote control device
- application services: a set of resources and services, such as compute power and applications
- network 110: the physical medium which connects devices and services
- lookup service 120: a database of network services

E. Architecture

As noted above, the palm sized computer 100 functions as the remote control device for the PowerPoint presentation. It is a multi-function control device in that it can control a host of resources accessible via the network. The palm sized computer 100 accomplishes this via middleware (e.g. Jini) and a generic control protocol capable of issuing control commands to an offboard resource. The palm sized computer 100 implements this control via the software components depicted in Figure 2. These software components represent a generic architecture for *control* of any network-based resource using a reduced functionality computer such as a palm sized computer. The software architecture, middleware, and control protocol implement a new model for lightweight mobile computing. This model of lightweight mobile computing is particularly well-served by devices such as a palm sized computer, as they possess the requisite size and portability.

In Figure 2, the control device 200 is able to control services on the network 110, such as the network based service 250. The directory of services 220 provides the registry functions used to enable the system.

The control device 200 includes a control device operating system 201 which supports a network communications program 202 and a control application 210. The network

communications program 202 allows the control device 200 to communicate with other devices on the network. The control application 210 includes the following elements:

- a GUI 212 to display the available network-based services and accept user input
- an application control protocol manager 214 which interfaces between the control device 200 and the network based computer service 250 by requesting tasks from that service (e.g. slide manipulation). This module is responsible for generating the application control protocol to command the selected service.
- a middleware protocol manager 216 to transfer data between the control device 200 and the directory of services 220 (e.g. communication between Palm platform and the Jini Lookup). This module is responsible for generating the syntax necessary to communicate based on the type of middleware being used.

The service device operating system 251 includes a service device communications program 252 and a service control application 260. The service device communications program 252 is responsible for communicating with the network 110. The service control application 260 includes the following components:

- a service application control protocol manager 262 that interfaces between the network based computer service 250 and the control device 200 and accepts control tasks (e.g. slide manipulation) issued from the control device 200. This module is capable of sending back a response (e.g. status) to the control device 200.
- a middleware protocol manager 266 to transfer data between the network based computer service 250 and the directory service (e.g. communication between the application service 130 and the Jini Lookup). This module is responsible for generating the syntax used to communicate based on the type of middleware being used.
- an application 268 to perform functions on the network. The application 268 can be a desktop application that would not execute on a palm sized computer.

Returning to the specific example of the PowerPoint presentation, Figure 3 illustrates the architecture that could be used to implement such a system. In Figure 3, the control device 200 has been replaced with a palm sized computer 100 executing the Palm OS 301. The GUI 212 is specifically for PowerPoint control (see Figure 1 for an example of such a GUI). The Java Virtual Machine 318 is executing on the palm sized computer 100 and replaces the middleware 218. The middleware protocol manager 216 supports Jini discovery, lookup and

download protocols. PowerPoint control commands are issued to the network based computer service 250. The PowerPoint control protocol manager 364 provides the interface for these commands and controls the PowerPoint application 368.

F. Control Flow

The process for accessing and controlling network-based services is described below. The specific implementation is control of a PowerPoint Service via a 3Com Palm Computing platform is used as an example.

1. Configure a computer hosting the directory of services 220 and connect it to the network 110. For instance, a Jini lookup service is configured to listen at a preset TCP, or UDP, port for service registration or service lookup requests.
2. Establish a network connection from the network based computer service 250 to the network 110.

30 Register the computing service with a directory of services 220. For example, in the Jini model, the computing service initially sends out a multicast packet announcing its presence on the network. Once the directory service 220 receives the packet, it sends a unicast packet back to the announcer. The packet includes an interface (e.g. Java code) for uploading code back to the directory service 220 as well as for searching and downloading code from the directory service 220.

40 Upload service description to the directory service 220. If Jini is adopted as the middleware, the application service 130 receives the response from the directory service 220 and uses the included interface to upload its service interface to the directory service 220. When the service interface is called, it contacts the directory service 220 which in turn creates an entry (represented by an object) for this new service and sets the proper fields such as service name, attributes and optionally the corresponding service interface. Other middleware may choose to use protocol-based approach such as FTP or TFTP for the uploading process.

50 Register the storage service 150 and display service 140 via the same process.

60 Establish a network connection from the control device 200 to the network 110. For Palm computers, there are multiple options for network connectivity. Possible solutions include using the infrared (IR) port to talk to a IR-LAN bridge or router, using the serial port to talk to a serial-to-LAN bridge or router, using either the IR or the serial port to talk to a digital cell phone and dial up a modem server, and/or using wireless data communications.

70 Launch the service control graphical user interface (GUI) 212 on the control device 200.

80 Via the service control GUI 212, accept user input, such as the selection of an application (for example, a PowerPoint application) to be controlled.

90 Optionally register the control device 200 with the directory service via a registration protocol, such as the Jini Discovery Protocol. This step is the same as the above one for the other services. It is executed only if the control device 200 has resources to offer.

100 Search the directory service 220 and download the desired service descriptor. In the case of Jini, after the control device 200 receives the response from the directory service 220, it uses the included interface to search the directory service 220 for an application service using the object type representing the service (such as an object type of PowerPoint presentation service) and the desired service attributes (such as the name and the physical location of the service). Once the directory service 220 finds such a service entry, the control device 200 is notified, which in turn uses the downloading interface to download the application service descriptor. One example of these services is the GUI code for controlling a PowerPoint presentation.

110 Send requests from the control device 200 to the network based computer service 250 to control the desired application. For example, a "next slide" request could be sent from a palm sized computer 100 to an application service 130 running PowerPoint. The communication can be based on a protocol such as the following one:

Control Type | Application | File Name | Control Function

Where: Control Type = {Request, Reply}

Application = {PowerPoint, FAX, Print, Email, Phonebook, ...}

File Name = {3ComPalmVIISpecification}

Control function = {File Load, Slide Forward, Slide Backward, File Close, ...}

Alternatively, techniques such as Java's Remote Method Invocation (RMI) can be used to achieve the same goal. In this case, the control device makes a local function call such as doForwardSlide(). The RMI mechanism will transfer the call to a remote machine which implements and carries out the function call. The PowerPoint presentation service may in turn

use other services such as the storage service 150 and the display service 140. The procedure to employ these services is similar to steps 10-11 above.

120 Accept any response to requests sent from the control device 200 to the network based computer server 250 and process any errors.

G. Control Device GUI

An important element of the control application 210 is a GUI front-end which accepts user input for controlling the PowerPoint presentation (or other application) and a control protocol manager backend which takes user input and translates it into commands to the CPU service. An example GUI is depicted in Figure 1. The example GUI allows the user to click on "forward", "backward", "go-to-first-page" or "go-to-last-page" buttons to control the slide show. The user can also click a "get-list" button to get a full list of the slide titles in the current presentation and choose to go to a particular slide. By clicking the "scribble" button, the window switches to graphics mode. In this mode, the user can draw at random on the panel, and the result of the drawing will be sent to the CPU service and eventually displayed on the projection service.

As explained in the software flow section, there are multiple ways to implement the application control protocol manager 214, the following illustrates one approach. The protocol takes the following form:

Control Type	Application	File Name	Control Function

Where:

Control Type = {Request, Reply}

Application = {PowerPoint, FAX, Print, Email, Phonebook, ...}

File Name = {3ComPalmVIISpecification}

Control function = {File Load, Slide Forward, Slide Backward, File Close, ...}

When the user clicks any button or draws something on the GUI, the application control protocol manager 214 generates the corresponding field in the protocol and sends a command to the network based computer service 250 via a TCP/IP channel.

H. Service Control Application Design

Figure 4 illustrates a detailed software architecture for the service control application of Figure 3. The architecture includes three main elements: a CPU service 410 (corresponding to the application service 130), a storage service 420 (corresponding to the storage service 150) and a projector service 430 (corresponding to the display service 140). Each of the elements include middleware protocol layer management modules. Each module has a corresponding service application control module (e.g., PowerPoint Control Protocol Manager). To control specific network services, instances of those services are instantiated and are used by the corresponding control protocol manager. The following describes examples of such classes that can be instantiated for specific network services.

PowerPoint Control

The PowerPoint control, through OLE automation, instantiates and controls an instance of PowerPoint application. The PowerPoint control also communicates with the storage service 420 to store/retrieve presentations and the projector service 430 to view slides.

The PowerPoint control has the ability to have many presentations open at the same time and is capable of switching between presentations. A presentation has a collection of slides in it. Once open, the PowerPoint control allows easy traversal of the slides either by commands like previous, next slide or by direct access (e.g. slide number or slide title). When any change in slide position occurs, the PowerPoint application automatically generates the image that needs to be shown by the projector control and invokes the method on the projector control to show the changed slide.

Once a presentation is open, the PowerPoint control can have facilities such as *Add Comments* and *Add Scribbled Graphics* to the current slide. Additionally, it can allow the adding of new slides to the presentation.

Figure 5 illustrates an example object class hierarchy for the PowerPoint control class. The following describes the elements of Figure 5 in more detail.

PowerPoint

Property/Method Name	Type	Description

Presentations	Collection	List of open presentations

Presentation

Property/Method Name	Type	Description
Slides	Collection	Array of slides in presentation
Location	Storage	Location where presentation loaded from
View	Projector	Projector where presentation is viewed

Slide

Property/Method Name	Type	Description
Title	String	Title for the slide
AddComments	Method	Add given comments to slide at given co-ordinates
AddScribble	Method	Add given graphics to slide at given co-ordinates

Storage

Property/Method Name	Type	Description
Host	String	LDAP Server host where the presentation stored
FileName	String	File name and other details for presentation

Projector

Property/Method Name	Type	Description
Host	String	Projector host where slides shown
ShowSlide	Method	Show slide image on projector

LDAP Database Control

The LDAP database control provides file system services to store and retrieve presentations. This control gets commands from the PowerPoint control through the LDAP protocol, to search for presentations and return presentations.

An object class hierarchy for CLdapDB could include CLdapDb having to a specific storage system reference.

Storage

Property/Method Name	Type	Description
FileFilter	String	File filter used to get file list
FileList	Collection	List of files found on storage

Projector Control

The projector control provides image viewing services and has a simple image viewer that shows the image on a projection screen, monitor, display device or canvas. The control gets commands from the PowerPoint control. Specifically, to display images, the projector control could cause frames generated by the PowerPoint control to be displayed at the device controlled by the projector control. Other embodiments use more sophisticated techniques for displaying the PowerPoint information (e.g., support windowing system API calls that an application may make).

An object class hierarchy for CProjector could include CProjector having to a specific image view reference.

ImageView

Property/Method Name	Type	Description
FileName	String	File name to be displayed
View	Method	Show the image on the projector
Mode	Long	FullScreen mode

I. Directory of Services Design

The directory of services 220 encodes the set of services available on the network 110. The directory of services 220 describes the characteristics of these services and provides a means to locate those services. To illustrate this concept, an object-oriented directory service is used. The directory service will have objects whose attributes describe the features of available services and optionally include either code to invoke those services or a reference to such code. A directory service will typically be one of several components offered in the middleware a control device 200 will use.

Figure 6 illustrates an object class hierarchy which models several network-based services. These network-based services are Application Service, Operating System Service, Storage Service, Projection Service, and Service Location Service. Examples of each of these services are now given.

Define-Class Service
    Superclass: root
    Attributes:
        Name
        Status

Define-Class NetworkService
    Superclass: Service
    Attributes:
        Location
        PhysicalAddress

PhysicalMachine
Owner
Vendor
Version

Define-Class ApplicationService
Superclass: NetworkService
Attributes:
CodeLocation
SerialNumber
SupportedFeatures

Define-Class OperatingSystemService
Superclass: NetworkService
Attributes:
RealTimeOS?

Define-Class StorageService
Superclass: NetworkService
Attributes:
DatabaseType
Schema

Define-Class ProjectionService
Superclass: NetworkService
Attributes:
Resolution

Define-Class ServiceLocationService
Superclass: NetworkService
Attributes:

QueryProtocol

Schema

Figure 7 illustrates a partial object class hierarchy which models Application Services, including the PowerPoint Application Service:

```
Define-Class SlidePresentationService
    Superclass: ApplicationService

Define-Class GroupwareService
    Superclass: ApplicationService

Define-Class EmailService
    Superclass: ApplicationService

Define-Class SpeechRecognitionService
    Superclass: ApplicationService
```

Based on this object class hierarchy, objects which represent network services can be defined. These objects are stored in a directory of services 220, such as a Jini Lookup.

An object to instantiate a SlidePresentationService could look as follows:

```
Make-Instance SlidePresentationService
    Name "PowerPoint"
    Status "Active"
    Location "3Com Intranet"
    PhysicalAddress "Building300.Floor2.Cube323"
    PhysicalMachine "PowerBook G3"
    Owner "Elaine Lusher"
    Vendor "Microsoft"
    Version "98"
```

CodeLocation "system/applications/office/powerpoint:"
SerialNumber "169-43-4666"
SupportedFeatures "Scribble"

Other examples of objects which represent application services include:

Make-Instance SpeechRecognitionService
Name "Naturally Speaking"
Status "Active"
Location "3Com Intranet"
PhysicalAddress "Building300.Floor2.Cube100"
PhysicalMachine "Solaris"
Owner "Wenjun Luo"
Vendor "Dragon Systems"
Version "4.1"
CodeLocation "system/applications/research/dragonspeech"
SerialNumber "157-89-4323"
SupportedFeatures "Dictation for Microsoft Word"

Make-Instance GroupwareService
Name "Alta Vista Forum"
Status "Active"
Location "3Com Intranet"
PhysicalAddress "Building300.Floor2.Cube220"
PhysicalMachine "Windows NT"
Owner "Paul Huard"
Vendor "Microsoft"
Version "3.1"
CodeLocation "http://3Community/code/groupware/latest
SerialNumber "444-56-7777"
SupportedFeatures "Virtual Chat Room"

Make-Instance EmailService
Name "Netscape Mail"
Status "Active"
Location "3Com Intranet"
PhysicalAddress "Building300.Floor2.Cube300"
PhysicalMachine "Solaris"
Owner "Rick Nottingham"
Vendor "Netscape"
Version "5.0"
CodeLocation "http://3Community/code/email/latest
SerialNumber "456-34-6786"

If middleware resides on a proxy device (rather than on the control device 200), the control device 200 will need to locate such a proxy service. An Operating System Service class can encode services such as a JVM Service, a Linux Service, or a Jini proxy.

Make-Instance OperatingSystemService
Name "Jini Proxy"
Status "Active"
Location "3Com Intranet"
PhysicalAddress "Building300.Floor2.Cube120"
PhysicalMachine "Solaris"
Owner "Rick Nottingham"
Vendor "Sun"
Version "5.0"
RealTimeOS? "no"

Finally, directory services 220 (such as the Jini Lookup) are modeled. This could be encoded in a class such as the Service Location Service.

```
Make-Instance ServiceLocationService
    Name "Jini Lookup"
    Status "Active"
    Location "3Com Intranet"
    PhysicalAddress "Building300.Floor2.Cube150"
    PhysicalMachine "Solaris"
    Owner "Rick Nottingham"
    Vendor "Sun"
    Version "5.0"
    QueryProtocol "Jini Lookup Protocol"
    Schema "Service Directory Schema 1.0"
```

The present invention integrates a number of services running on different machines, and provides control of the services from a thin platform with the a graphical user interface controlled by a console application adapted to the thin platform. A typical single station-based application is organized into a set of modular services running from different systems all networked together. In one embodiment described with reference to Figures 8-16 below, the Powerpoint application was chosen and a set of control services were developed to control the various aspects of presentation of a slide. A wide variety of other applications are also controlled using the present invention, and some examples are mentioned herein. The details provided below concerning the Powerpoint application will be modified in various implementations of the invention and for various applications other than Powerpoint, and are provided only as an example implementation. In this example, all these control services where then controlled from the a palm sized computer running the Palm computing environment.

In the example described, the system consists of the following parts:

1. The PalmDOS (Palm Directory of Services) and Powerpoint control program running on the Palm.
2. PalmListener - The Java based server to integrate the Palm with the network.

3. The Powerpoint control Services.
4. The Jini lookup and registry service.

All the services register themselves to the Jini registry service as soon as they are started. When a user taps the PalmDOS service on the Palm screen, the PalmDOS service accepts the IP address of the PalmListner as input and contacts it. Alternatively, a discovery message protocol can be exectured to automatically find the PalmListener. The PalmListener in turn contacts the Jini server and passes on to the PalmDOS a list of services that are available. When the user taps the Powerpoint Presentation Service on the Palm the PalmDOS contacts the PalmListener. The PalmListener then sends a stream of Palm executable bytes and the IP address/port details of the Powerpoint control service to the Palm. The PalmDOS service installs the Powerpoint control program (from the byte stream) and Launches it after passing the IP address/port details as a launch parameter. The Powerpoint control program then interacts with the Powerpoint control service to control the presentations. The Powerpoint control program offers a UI to control presentation slides in a limited way.

The PowerPoint control services is a collection of services which facilitate the control and display of a PowerPoint presentation in a networked environment.

The PowerPoint control services aim to break down the various tasks in a typical single station based presentation of slides using PowerPoint application, into a set of modular services running from different systems all networked together.

Some of the tasks that are performed by a presenter of a presentation include:

- Locating and loading a presentation from a storage subsystem into PowerPoint application.
- Once loaded, scroll through the presentation, one slide at a time, doing tasks like annotating/scribbling on the slide, along the presentation.
- Display the slides on a wide screen through a projector connected to the PC running the PowerPoint application.
- Print the slides as needed.

The PowerPoint control services simulate these tasks using the services shown in Figure 8, all interconnected among each other as needed: Console, Cpu, Storage, Display, Print.

- The presenter runs the Console Application 800, which through its interface lets him/her control all aspects of giving a presentation. The Console Application also controls the behaviour of other services.
- The Storage service 801 provides a listing of all presentations available to be presented, that are stored in the subsystem on which it is running.
- The Display service 802 provides the facility to display the slide images of a presentation on a display medium.
- The Print service 803 provides the facility to print the slide images of a presentation on a print medium.
- The Cpu service 804 provides the necessary interfaces to communicate with the PowerPoint application to actually facilitate loading a presentation and showing/printing the slide images on the connected display and print services.

The PalmDOS and Powerpoint control program run on the Palm. The PalmDOS application basically connects to the PalmListener and gets a list of service and displays it to the user. When the user taps Powerpoint Control the PalmDOS downloads the executable version of the Powerpoint Control program and installs it on the Palm. It then launches the Powerpoint control program after passing the IP address /port details of all the powerpoint presentation services. ThePowerpoint control program presents a user interface for the user to connect to and control the presentation slides present in the Storage service and display it in the Display Service.The Palm does all network communication by means of establishing a Point to Point Connection Protocol (PPP) connection with a proxy. TCP/IP packets are sent over the PPP. A more detailed explanation of the Powerpoint control program is provided later. Please refer the appendix for details about the PPP connection between the Palm and Windows 95 and Windows NT machine.

General Overview Jini technology:

The JINI registry service was built using the Sun Microsystems Jini Technology. Jini Technology is a breakthrough initiative based on JavaTM technology that enables all types of devices

to simply connect into impromptu networks, making access and delivery of new network services as simple as plugging in a telephone. Built on top of a Java software infrastructure, Jini technology enables all types of digital devices to work together in a community put together without extensive planning, installation, or human intervention. Each device provides services that other devices in the community may use. These devices provide their own user or programmatic interfaces, which ensures reliability and compatibility.

The lookup service in Jini technology is a lightweight, but very powerful repository of services. Its structure is unique in that it uses the Java platform type system as the namespace. This means it does not store fixed name-value pairs, but objects and object graphs -- the actual behavior of an object. This has two distinct advantages in a distributed system. First, you can search for an object or service based on a desired behavior, not just on its name. Second, once you find an object in the directory, you immediately know how to use it .

ENVIRONMENT

The services are developed as Java classes in one embodiment, enabling them to be run on any Java supported platform.

Appropriate JNI interfaces enable the controlling of PowerPoint and other platform specific applications from a Java class. Tools for development and test of the Powerpoint example described herein are set forth in the following table.

J. Development & Test Environment

SOFTWARE	PURPOSE
Visual Café 3.0JDK1.2	Java classes for the various services.
PowerPoint and Office 97with Service Pack 1	Application controlled by the Cpu service.
Visual Studio 6.0	JNI interfaces on the PC.

Personal Web Server	Application that the storage service depends on when running on the PC.
Internet Explorer 4.0	Application that the display service depends on when running on the PC.

The PalmDOS application is mainly used to connect with the PalmListner and get a list of services and present to the user. Once the user taps on the service desired, the PalmDOS function downloads a byte stream containing a Palm executable file of Powerpoint console program. It also gets a List of addresses to the locations of the various Powerpoint Control services.

As soon as the palmDOS service obtains the Byte stream of the Powerpoint control program application it installs the Powerpoint control program on the palm and launches it. It also Passes the list of addresses as a command line argument to the Powerpoint control program.

Once the user enters the IP address of the PalmListener service in the field provided and taps the OK button the PalmDOS connects to port 2300 (or some other specified part number) of the PalmListener. Which in turn spawns a child to handle the PalmDOS request. The communications are handled in such a manner that for each request the PalmDOS creates a new socket and reconnects to port 2300 of the PalmListener. This design helps the PalmListener to manage multiple Palms connecting to it at the same time.

The Powerpoint control program Application is the main interface through which the collection of services is controlled to facilitate giving a presentation. It offers the user the list of the various service locations and the user can choose from the list.

2.0.1 Command :Get a list of Services registered to the JINI

The following command is issued by the PalmDOS to the PalmListener to get a list of services.

SENDRQ

2.0.2 Command : Download Powerpoint control program Byte stream

The following command is issued by the PalmDOS to the PalmListener to get the Powerpoint control program byte streams and the list of addresses of the powerpoint control services. Using the touch screen on the palm, the user executes a :

Tap on the name of the service on the List

Console Application

The Console Application is the main interface through which the collection of services are controlled to facilitate giving a presentation. Console Application is the only service in the collection that has an user interface. All the other services may provide minimal user interface required to setup the programs.

Since the Console Application is expected to run from a palm sized computer, such as the Palm Computing Environment available from 3Com Corporation, the console application service is expected to use least number of resources in terms of network connections and features like views and windows.

The Console Application functionality can be described using the following main tasks:

1. User interface
2. Collection of information about the location of various services that will participate in the presentation.
3. Communication with the Cpu service to carry out the presenter's commands.
4. Opening a presentation and navigation of slides in a presentation.
5. Editing/Annotating/Scribbling on a slide during the presentation.
6. Display/Print the image for current slide on a connected Display/Print service.

K User Interface

The Console Application is the only service in this example which has an user interface. It is started in this example by selecting the PalmDOS icon in the screen of Figure 9.

There are three distinct stages to the user interface:

- Collection of information about service location
- Navigation of slides
- Editing/Annotating/Scribbling on a slide

L. Service Location

When started, the Console Application presents the user with a choice of system addresses (IP address/host name & tcp port no) where the various services are configured to run. It can connect to a single Cpu service and a single Storage Service and multiple Display and Print services. The PalmDOS program then shows the screen of Figure 10 in one example.

In one implementation, the location service provides a hardwired list of system names with appropriate port numbers where the various services are expected to be configured and run (Fig.11). In another implementation, the user interface on the palm sized computer, or other console application host, provides an icon, which when selected invokes a discovery message protocol to access a registry of available services, and produce the list based upon information gathered at the site served by the registry which responds to the discovery message. The PalmDOS application provides the system addresses as command line parameters when the Powerpoint console application is launched.

M. Cpu Communication

Once the various hosts/locations have been selected, Console Application connects to the Cpu service and informs it about the various other choices. Even though the Console lists locations for all available Storage/Display/Print services, it does not directly communicate with any of them. Instead, it lets the Cpu service establish links to the selected services and communicate with them as needed based on the task to be performed.

Once connected, the Cpu service in turn tries to connect to the Storage, Display and Print services. The list of presentations available on the storage subsystem is then sent back to the Console to be displayed. The user can select a presentation from the list, that needs to be presented.

The Console remains connected to the Cpu until all the services are stopped. The Cpu service acts as a gateway between the Console and all other connected services.

Most of the commands that are used in this example have one or two letter codes with optional parameters following the command code. The command typically results in appropriate response.

2.1.1 Command: Location of Display Service

The following command is issued by the Powerpoint control program Application to Cpu, to specify the location a Display service:

P<hostname>;<portno>

Where p is the command code, *<hostname>* is the name of the host where the display service is running and *<port>* is the port no where the display service is bound.

On receipt of this command, the Cpu service establishes a socket connection to the display service at the given location.

2.1.2 Command: Location of Storage Service

The following command is issued by the Powerpoint control program Application to Cpu, to specify the location of the Storage service:

l<hostname>;<portno>

Where l is the command code, *<hostname>* is the name of the host were the storage service is running and *<port>* is the port no where the storage service is bound.

On receipt of this command, the Cpu service establishes a socket connection to the storage service at the given location.

After establishing the connection, Cpu sends the following command to the connected storage service:

l

where l is the command code which lists all files known to the storage service. The storage service lists the file names with appropriate url names as configured with the web server which is running along side the storage service.

The list of presentation names is then returned back to Powerpoint control program to be displayed.

2.1.3 Command: Location of Print

The following command is issued by the Powerpoint control program Application to cpu, to specify the location of the Print service:

Ph < hostname > ; < portno >

Where Ph is the command code, < hostname > is the name of the host where the print service is running and < port > is the port no where the pritn service is bound.

On receipt of this command, the Cpu service establishes a socket connection to the print service.

2.1.4 Command: Print current slide

The following command is issued by the Powerpoint control program Application to Cpu, to print the current slide (of a presentation or blank page which is being scribbled on):

Ps

This command must be given only after establishing a connection to the print service. On receipt of this command, the Cpu service copies the image of the current slide to the print service which will then using IE, print the image on currently selected printer.

2.1.5 Command: Print all slides

The following command is issued by the Powerpoint control program Application to Cpu, to print all slides in current presentation:

Pa

As with Ps, this command must be given only after establishing connection with print service. This command is valid only when a presentation is open and not in blank slide mode.

2.1.6 Command: Open Presentation

The following command is issued by the Powerpoint control program Application to Cpu, to open a presentation. The name of the presentation is selected from the list of names returned earlier after locating the storage service.

o *<presentation>*

2.1.7 Command: Close Presentation

The following command is issued by the Powerpoint control program Application to Cpu, to close an open presentation:

c

This is done so that at the maximum only two presentations will be open at any time (the presentation opened explicitly and the scratch presentation used for blank screen).

2.1.8 Command: Save Presentation

During the presentation, if changes are made to a presentation, it is possible to save the changes back on the storage. The save presentation could also be used to save the scribbled slides.

Clicking the Save button, instructs the Cpu service to save the current presentation on the storage service. If the current mode is blank screen mode, then the scribbled blank slides are saved on the storage service (using the timestamp as file name; like mm-dd-yyyy.hh.mm.ss.ppt).

The syntax for the save command as sent to the Cpu service is:

v

This Cpu responds to this command with the Saved message. Depending on the current mode, the appropriate presentation is saved by the storage service.

2.1.9 Command: Get Titles

After opening a presentation, the Powerpoint control program Application gets a list of titles from the current presentation. This is achieved in two steps:

Get title count
Get title for a given slide

2.1.9.1 Command: Get title count

To get the number of titles that are there in a presentation,Powerpoint control program sends the following command to Cpu:

Gc

This results in a count, which is same as the number of slides in the presentation.

2.1.9.2 Command: Get title for a given slide

To get the title for a specific slide in the current presentation,Powerpoint control program sends the following command to Cpu:

G<idx>

Where <idx> is the slide no (which is between 1 and last slide no).

2.1.10 Command: Get Text

Whenever a slide is navigated to (either by clicking on a title, or using the navigation buttons), the Powerpoint control program Application gets the text from the current slide and displays it. This is done so that the view from the presenter is in sync with what is displayed at all the connected display services.

Like the get title command, this command is also done in two steps:

Get text count
Get text for given textbox

2.1.10.1 Command: Get text count

To get the no of text boxes in current slide,Powerpoint control program sends the following command to Cpu:

gc

This returns the count of text boxes: in most slides, this will be 2; 1 for the title and other for the outline body text.

2.1.10.2 Command: Get text for a given text box

To get the text from a given text box,Powerpoint control program sends the following command to Cpu:

g<idx>

This returns the text for the given text box. In most slides, there will be only 2 text boxes (actually, in PowerPoint object model lingo, these are referred to as placeholders, one being a title placeholder and other being text (outline) placeholder).

Cpu formats the body text, if indentation-using bullets are present. To show the outline of text properly for various indentation levels, Cpu uses the following convention:

. <level 1 indent>
- <level 2 indent, with 2 leading spaces>
+ <level 3 indent, with 4 leading spaces>
* <level 4 indent, with 6 leading spaces>
> <level 5 indent, with 8 leading spaces>.

1.0.1 Command: Get speaker notes

To get the text of speaker notes, which are part of the current slide,Powerpoint control program sends the following command to Cpu:

N

This returns the speaker notes (if found). No special formatting is done to the text found as speaker notes. This text is not editable.

1.1 Slide Navigation in Powerpoint control program application

Once the user opens a presentation (from the list of presentations of Fig.15 for example), the Powerpoint control program sends a command to Cpu service to open the presentation. This sets off a sequence of commands that finally results in the presentation being loaded in to

PowerPoint and the first slide shown on all displays. During the presentation, the Powerpoint control program remains in sync with what is shown on the displays.

The Powerpoint control program shows only the text from the current slide, whereas the display shows the complete image as would be shown if the presentation were given directly from inside PowerPoint.

The Powerpoint control program provides controls for easy navigation of slides in a presentation:

- A list of titles in the current presentation is shown in a separate Form like Fig.16 for example. Clicking on any slide, results in a command to Cpu to go to that slide. This in turn results in the image for the slide shown on all connected displays. The window, which shows the text of the current slide on the Powerpoint control program, is also updated with the text from the just navigated slide.
- Buttons are provided to go to: first slide, last slide, next slide and previous slide.

2.0.1 Command: Goto slide using title

When the user selects a title to scroll to, the Powerpoint control program Application issues the following command to Cpu service:

t <*title-text*>

where <title-text> is the title as it appears on a slide. If a slide does not have a title, the '-' is substituted for the title.

2.0.2 Command: Goto first slide

When the user clicks this button, the Powerpoint control program issues the following command to cpu service:

<<

Cpu service makes the first slide as the current slide and exports the image to display services.Powerpoint control program gets the text for the first slide and shows it.

2.0.3 Command: Goto next slide

When the user clicks this button, the Powerpoint control program issues the following command to cpu service:

>

Cpu service moves to the next slide in the presentation. If already in the last slide, then nothing happens.

2.0.4 Command: Goto previous slide

When the user clicks this button, the Powerpoint control program issues the following command to cpu service:

<

Cpu service moves to the previous slide in the presentation. If already in the first slide, then nothing happens.

2.0.5 Command: Goto last slide

When the user clicks this button, the Powerpoint control program issues the following command to cpu service:

>>

Cpu service moves to the last slide in the presentation.

5

2.1 Edit Text/ Scribble on a slide

After opening a presentation and during navigating the slides, the user may do the following tasks on the current slide:

- Edit the text on the slide.
- Scribble on a blank slide.

2.0.1 Edit text

During the presentation, the user can change the text on the current slide. An ***Apply*** Menu item is provided which when clicked, sends a command to the Cpu to apply the text changes to the current slide. As with navigation of slides, when any changes are made to the slide, the Cpu instructs the Display service to display the changed slide's image.

While editing the text, the user can easily simulate the indented (bulleted) structure of body text using the same notation that Cpu uses to illustrate the outline text. Unlike the formatting done in Cpu, in the Powerpoint control program, no additional leading space needs to be specified; just the appropriate first letter to signify the indentation (bullet) level.

.	Level 1 indent
-	Level 2 indent
+	Level 3 indent
*	Level 4 indent
>	Level 5 indent

3.0.0.1 Command: Change the text on slide

The Powerpoint control program Application shows the text from various text boxes in the slide as strings in a text area, as in Fig.12 for example. The text from each text box is delimited by tilde character (`). The user can change the text within the delimiter and apply the changes onto the slide using pull down menus, like shown in Fig.13.

When the user modifies the text in a slide, the Powerpoint control program issues the following command:

S<idx>;<slide-content-text>

The slide content text is the entire text for the slide as changed by the user. The Cpu service in turn applies the changes on to the slide. This command will only change the text on the slide; it will not be exported into an image and shown on displays; To do that the Apply command needs to be issued.

The <idx> parameter can have one of two values: if set to 1, the following text includes the title text for the slide and the outline body text. These two texts are separated by (`) character; if set to 2, the following text is the changed text for the speaker notes.

3.0.0.2 Command: Apply changes and refresh display

After making the changes, to apply the changes on to the slide and show it on displays, the Powerpoint control program issues the following command:

A

This instructs the Cpu service to apply all the changes on to the slide and export the image onto all connected display services.

3.0.1 Scribble

During the presentation, the user can request for a blank slide and scribble on it using free form strokes on a predefined area of the Powerpoint control program Application's window as shown in Fig.14. A ***Blank*** menu item is provided which toggles the blank slide and normal slide mode.

Clicking the Blank menu item, instructs the Cpu service to use a blank slide and show it on the Display services. In sync with the Cpu and display services, the Powerpoint control program

also shows a blank painting area. The presenter is then allowed to scribble on this area and the strokes are conveyed to the Cpu and in turn to the display services. Clicking on the Unblank button toggles the mode back to text mode and the last slide that was being shown is switched back as the current slide (the Cpu and display services also switch their contexts to the correct slide).

3.0.1.1 Command: Scribble

The syntax for the scribble command as sent to the Cpu service is:

s<cnt-nodes>;<xynode1><xynode2>...

where the first parameter is the number of nodes in the scribble and the actual nodes represented as x and y coordinates.

3.6 Functional Flow of the Powerpoint Control Program

There are five different forms in the Powerpoint Control program. Each form has a handler to handle the events generated in that form. Following are the forms and their corresponding handlers.

Main Form (showing the list of addresses): MainFormHandleEvent
Title Form (showing the text for the slide): TilteFormHandleEvent
Title Table form(showing the list of titles) : TitletableHandleEvent
File Form(showing the list of files) : FileFormHandleEvent
Scribble Form(blank form to scribble): ScribbleFormHandleEvent

TitleFormInit is called ti initialise the Title form and is called whenever the Title Form is re-entered. GetText is called when the navigation buttons are pressed. The function GetTime is called whenever the clock field is to be updated.

The Send and Recv funcations are called whenever a network send and receive is required.

Finally the StopApplication is called to close all the sockets and the network library.

3 Command Summary

3.1 PalmDOS Command Summary

Command	Description
SENDRQ	Send the list of services currently registered
Powerpoint Control Services	Send the byte stream of Powerpoint control program application and the list of addresses of the Powerpoint Control Services

3.2 Console Command Summary

The following table summarizes the commands that are received by the Cpu service from the Powerpoint control program Application; these commands have been briefly explained in the section on Powerpoint control program Application:

Command	Description
p<hostname>;<port>	Display service host and port address
l<hostname>;<port>	Storage service host and port address

Ph < hostname > ; < port >	Print service host and port address
Ps	Print current slide
Pa	Print all slides (not blanks)
o < url-presentation >	Open a presentation given the URL
c	Close currently open presentation
v	Save current presentation
Gc,G < idx >	Get titles from current presentation
gc, g < idx >	Get text for current slide
S < idx > ; < text >	Set text for current slide
A	Apply changes to slide and refresh
t < title >	Scroll to slide with specified title
< <	Scroll to first slide
> >	Scroll to last slide
<	Scroll to previous slide
>	Scroll to next slide
b	Toggle blank screen mode
s < nodes > ; < nodelist >	Scribble on blank screen using nodes
Q	Quit

CPU SERVICE

The Cpu service is the main gateway interface between Console and the other services like Storage, Display and Print. The Console does not directly communicate with Storage, Print and Display services; only through the Cpu service.

In this example for controlling the PowerPoint application and Office 97, the Cpu service is from a Windows 95/NT workstation. Also this service would require Java Native Interface (JNI) to another OLE automation module which actually would communicate with PowerPoint.

The Cpu service functionality can be explained based on the following tasks it performs:

1. Communication with Console Application
2. Communication with Storage service
3. Communication with Display service
4. Communication with Print service
5. JNI to PowerPoint automation module

N. Communication with Console Application

The Console sends commands to the Cpu service which responds to the commands and sends responses back. Almost all commands result in some form of response.

After startup, the first set of commands from the Console to the Cpu, establish the location of the various services like Storage and Display. Since the Console does not communicate directly with other services, Cpu establishes connection to these services and passes appropriate response back to Console.

The Console sends the following commands to the Cpu. Some of these commands need to given in a sequence (like navigation commands follow the open presentation command, which follows service location messages, etc.)

- Set service location for Display, Storage and Print services
- Get a list of presentations from Storage
- Open a presentation
- Get a list of titles in currently open presentation
- Go to a slide using the title
- Go to the first, next, previous or last slide

- Get the text content from current slide
- Apply changes to text on the current slide
- Add notes to current slide at given location
- Enable/Disable blank screen mode
- Apply scribble on current blank slide
- Display current slide on Display services
- Print current slide on Print services
- Shutdown

O. The Communication with Storage service

The Cpu service gets a list of presentations available on the storage subsystem and sends them back to the Console.

Even though the storage service makes a list of presentations available on the storage subsystem, the Cpu service is responsible for actually transferring the file from the storage sub-system to the machine where the Cpu service is running.

Storage Command Summary

The following table summarizes the commands that are issued by the Cpu service to the storage service:

Command	Description
l	Return list of presentations on storage
q	Quit

P. Communication with Display service

The Cpu service creates images of slides as they are being scrolled through and sends them to the display service to be shown on any connected display medium (like Video screen, projector etc.)

The communication between Cpu and Display is mostly one way and include:

- Cpu sends an image to display; Display shows it on the display medium. Multiple Display services could be connected to the Cpu, which would all receive images to be shown.
- Shutdown

Display Command Summary

The following table summarizes the commands that are issued by the Cpu service to the Display service:

Command	Description
<imagesize> <data>	Store the image and show it on display
f	Toggle full screen mode
q	Quit

Q. Communication with Print service

The print service is very similar to the Display service, the only difference being the output medium.

R. JNI to PowerPoint automation module

The PowerPoint automation module is responsible to translate the commands from Console, carry them out on the PowerPoint application.

These commands include:

- Open a presentation using the given local file name
- Close an open presentation

- Get a list of titles in current presentation
- Get the number of slides on current presentation
- Goto a slide based on the given title
- Goto a slide based on the slide no
- Goto first, next, previous or last slide
- Get text from all textboxes from current slide
- Apply text changes on current slide
- Add notes to current slide at given location
- Switch presentations between current open presentation and a temporary scratch presentation
- Switch between current slide on an open presentation to a blank slide on scratch presentation
- Apply scribble on a blank slide
- Create and export the image for current slide (from open presentation as well as from scratch) to some location

S. JniCpu class

The following code segment illustrates the typical implementation for the JNI for PowerPoint automation module.


```
public class Cpu
{

/**
 * StartPP ()
 *
 * Start the PowerPoint application. PowerPoint is a single instance application (only one
 * copy of it can be running on the system at a time. If an instance is already running, then it
 * will be used. Else a new instance is created and made visible. The instance is made visible
 * so that some of the scribble and export functions work properly.
 *
 * If this function fails, then rest of the methods can't be used.
 *
 * Input:
 *      Nothing
```

```
     *
     * Output:
     *     Nothing; PowerPoint is started.
     *
     * Returns:
     *     0      -  If PP started succesfully.
     *     != 0   -  Error codes, that explain the failure.
     */
    public native long StartPP ();

    /**
     * StopPP ()
     *
     * Quit the PP application. If any presentations are open, they are all closed.
     *
     * Inputs:
     *     Nothing
     *
     * Outputs:
     *     Nothing
     *
     * Returns:
     *     0      -  If PP stopped succesfully.
     *     != 0   -  Error codes, that explain the failure.
     */
    public native long StopPP ();

    /**
     * OpenPresentation ()
     *
     * Open a presentation. The file name given as input to this method is an absolute path that must
     * be accessible to the PP. This implementation only supports one open presentation at a time. If
     * you need to open another presentation, then first close the currently open presentation using the
     * ClosePresentation method and then invoke this method.
     *
     * Input:
     *     strFile -  Name of the presentation (.ppt file) to open
     *
     * Output:
     *     Nothing
     *
     * Returns:
     *     0      -  If given presentation opened
     *     != 0   -  Error codes, that explain the failure.
     */
    public native long OpenPresentation (String strFile);

    /**
     * GetExportFile ()
     *
     * Get the name of the export file (as a result of a command that results in a export of a slide
     * commands like scroll page, add notes, add scribble result in export).
     *
     * Input:
     *     Nothing
     *
     * Output:
```

```
     *      Nothing
     *
     * Returns:
     *      Name of the file, last exported.
     */
    public native String GetExportFile ();

    /**
     * ClosePresentation ()
     *
     * Close the currently open presentation.
     *
     * Input:
     *      Nothing
     *
     * Output:
     *      Nothing
     *
     * Returns:
     *      0       -   If presentation closed.
     */
    public native long ClosePresentation ();

    /**
     * GetCurrentSlide ()
     *
     * Get the current slide no in the currently open presentation.
     *
     * Input
     *      Nothing
     *
     * Output
     *      Nothing
     *
     * Returns:
     *      <no>    -   Current slide no 1 - max slide no
     *      -1      -   if a presentation is not open or other failure
     */
    public native long GetCurrentSlide ();

    /**
     * GetSlideCount ()
     *
     * Get the number of slides in the currently open presentation.
     *
     * Input:
     *      Nothing
     *
     * Output:
     *      Nothing
     *
     * Returns:
     *      <cnt>   -   No of slides in current presentation
     *      -1      -   If a presentation is not open, or other failure
     */
    public native long GetSlideCount ();
```

```
/**
 * GetTitles ()
 *
 * Get all the titles from the currently open presentation. This is a list of lines separated by
 * '\n'. If a slide does not have a title, then a blank line is returned.
 *
 * Input:
 *     Nothing
 *
 * Output:
 *     Nothing
 *
 * Returns:
 *     String -  Contains the titles.
 */
public native String GetTitles ();

/**
 * GetText ()
 *
 * Get all the text from the currently open presentation. This is a list of lines separated by
 * '\n'.
 *
 * Input:
 *     Nothing
 *
 * Output:
 *     Nothing
 *
 * Returns:
 *     String -  Contains the text.
 */
public native String GetText ();

/**
 * SetText ()
 *
 * Set the changed text on the current slide on the currently open presentation. This is a list of lines
 * separated by '\n'.
 *
 * Input:
 *     strText -  String containing the changed text
 *
 * Output:
 *     Nothing
 *
 * Returns:
 *     0      -  If change applied properly
 *     != 0   -  Error codes, explaining the failure.
 */
public native long SetText (String strText);

/**
 * GotoSlideNo ()
 *
 * Scroll to the given slide no. The no must be between 1 and max-slides in the presentation. If any display
 * services are registerd with the CPU service, then the CPU service would scroll to this slide and export it
```

```
     * and push the URL to the display services.
     *
     * Input:
     *     iNo    -  Slide no to scroll to.
     *
     * Output:
     *     Nothing
     *
     * Returns:
     *     0      -  Scrolled to the given slide successfully.
     *     != 0   -  Error codes explaining the failure.
     */
    public native long GotoSlideNo (int iNo);

    /**
     * GotoSlideDir ()
     *
     * Scroll in the given direction. Export the slide and push the URL to all known display services.
     *
     * Input:
     *     iDir   -  -1 scrollPrev
     *               -  -2 scrollNext
     *               -  -3 scrollFirst
     *               -  -4 scrollLast
     *               - -5 scrollCur - refresh the current slide
     *
     * Output:
     *     Nothing
     *
     * Returns:
     *     0      -  Scrolled to the given slide successfully.
     *     != 0   -  Error codes explaining the failure.
     */
    public native long GotoSlideDir (int iDir);

    /**
     * GotoSlideTitle ()
     *
     * Scroll to the given title. Figure out the slide no based on the text.
     *
     * Input:
     *     strTitle -  Title of the slide to scroll to.
     *
     * Output:
     *     Nothing
     *
     * Returns:
     *     0      -  Scrolled to the given slide successfully.
     *     != 0   -  Error codes explaining the failure.
     */
    public native long GotoSlideTitle (String strTitle);

    /**
     * BlankScreen ()
     *
     * Turn On/Off the blank screen mode. This will create a new blank slide on a scratch presentation
     * that is used to do the scribble.
```

```
     *
     * Input:
     *     bFlag  -  true   -  Blank screen
     *               false  -  UnBlank screen
     *
     * Output:
     *     Nothing
     *
     * Returns:
     *     0      -  A blank slide created successfully.
     *     != 0   -  Error codes explaining the failure.
     */
    public native long BlankScreen (boolean bFlag);

    /**
     * AddScribble ()
     *
     * Scribble on current blank slide using the give nodes. Each node represents a point on the slide. A line
     * is drawn from one point to the next till all nodes are drawn. The blank screen mode must have been
     * turned ON before making this call. After drawing the nodes on the slide, the page will be exported out  * and
    pushed out on to the known display services.
     *
     * Input:
     *     lNodes  -  No of nodes
     *     strNodes-  Nodes represented as a pair like "(xxxx,yyyy)"
     *
     * Output:
     *     Nothing
     *
     * Returns:
     *     0      -  If scribble added to the current slide successfully
     *     != 0   -  Error code explaining the failure
     */
    public native long AddScribble (long lNodes, String strNodes);

    /**
     * AddNotes ()
     *
     * Add a sticky notes to current slide at given location. If there are other notes at the
     * current location, then PP will automatically slide the notes down.
     *
     * Input:
     *     x,y    -  Location where the notes is to be shown.
     *     strNotes-  String containing the notes.
     *
     * Output:
     *     Nothing
     *
     * Returns:
     *     0      -  If notes added successfully
     *     != 0   -  Error codes explaining the failure
     */
    public native long AddNotes (int x, int y, String strNotes);

    }
```

T. CPowerPoint class

The following code segment illustrates the typical implementation for a PowerPoint automation class as implemented in C++. As can be seen, there is a direct one to one correlation between the JniCpu class and CPowerPoint class.


```
class CPowerPoint
{
public:
        CPowerPoint();
        virtual ~CPowerPoint();

public:
        // Start an instance of PowerPoint (if one already running, then use that)
        // since PP is a single instance app
        int StartPP ();

        // Stop (File/Exit) running PP
        int StopPP ();

        // make PP visible
        int Visible (bool bVisible);

        // open given presentation
        int OpenPresentation(_bstr_t bstrFile);

        // return the no of slides in presentation
        int SlideCount ();

        // return list of titles in presentation
        int GetTitles (_bstr_t *pbstrTitles);

        // goto a given slide (based on direction)
        int GotoSlide (enum ScrollValue dir);

        // goto a given slide (absolute slide no)
        int GotoSlide (int iNo);

        // goto a given slide (based on slide title)
        int GotoSlide (_bstr_t bstrTitle);

        // add notes to current slide
        int AddNotes(_bstr_t bstrNotes, float cX, float cY);

        // add scribble to current slide/new blank slide
        int AddScribble (bool fNewSlide, long lNodes, struct Point *pNodes);

        // get text from current slide
        int GetText (_bstr_t *pbstrText);

        // set the new text
```

```
        int SetText (_bstr_t bstrText);

        // get the size of the slides (width & height, in points)
        int GetSize (int *pWidth, int *pHeight);

        // Export current slide to jpg file
        int Export();

        int BlankScreen ();

        // close opened presentation
        int ClosePresentation ();

        // name of recently exported file
        _bstr_t m_bstrExportFile;

        long CurrentSlide () { return m_fBlank == false ? m_lSlideIdx : 1; }

        long FileSearch (_bstr_t bstrPath);

        FoundFilesPtr FoundFiles () { return m_pPP->GetFileSearch()->FoundFiles; }

private:
        // status flags
        cppFlags m_cppFlags;

        // main PowerPoint application pointer
        // use this to create an instance of PP
        _ApplicationPtr m_pPP;

        // currently open presentatio pointer
        // only one presentation is active and open at a time (for now)
        _PresentationPtr m_pPres;

        _PresentationPtr m_pBlankPres;
        bool m_fBlank;

        // cache slide titles
        _bstr_t *m_bstrTitles;

        // no of slides in pres
        long m_lSlideCnt;

        // current slide idx
        long m_lSlideIdx;

        // refresh titles
        void RefreshTitles ();
};

enum ScrollValue {
        scrollFirst = 1,
        scrollNext = 2,
        scrollPrev = 3,
        scrollLast = 4
};
```

STORAGE SERVICE

The Storage service is a gateway interface between the Cpu service and a web server. The Storage service is only responsible to collect a list of presentations available on the subsystem shared by the web server. It is not required to actually transfer the presentations to the Cpu service. That is done in coordination with the web server.

After starting up, the Storage service gets commands from the Cpu service, which include:

- Get a list of presentations on the storage subsystem and return the list in the form of URLs
- Shutdown

The URLs returned by the Storage must be created in such a way that when the Cpu service needs to load a presentation it is able to download the file from the web server which is also running on the same system.

DISPLAY SERVICE

The Display service is an output only type of service, in the sense that it does not communicate back results to any other service. Many instances of the Display services may be running on different systems, all of which will be connected to the Cpu service. The purpose of this service is show the image on a display medium, when instructed to do so.

After starting up, the Display service gets commands from the Cpu service, which include:

- Copy the image for a slide and show it on the display medium
- Shutdown

Whenever the context requires, Cpu service generates the image for the current slide being presented and instructs the Display services to refresh the display about the new image. The Cpu sends the entire image to the Display service which shows the image.

PRINT SERVICE

The Print service is an output only type of service and is very similar to the Display service, the only difference being the output medium. Unlike the Display service, the Cpu service will not refresh the image often; it will do so only on instruction from the Console Application that a print image is requested of the current slide.

When required, the Print service will print the image on the connected print medium.

After starting up, the Print service gets commands from the Cpu service, which include:

- Copy the image for a slide and print it on the print medium
- Shutdown

U. Variations on the Network-based Control Application

Several variations of the general control paradigm can be defined:

- The middleware (Jini, in our example) may not be physically resident on the control device 200. In this case, a proxy is used which runs middleware on behalf of the control device 200. Functionally speaking, the same design will still apply. One change to the design can be used to support a modified GUI for the control device. In these embodiments, the GUI program is modified by the proxy device to account for specific limitations of the control device. Also, in some embodiments, the control device may include preset applications that interface directly with the proxy device. Such an architecture would support a limited set of network services but would likely result in very small and lightweight applications on the control device.

- The control paradigm is not limited to palm sized computers. Any computing device with restricted computing power could be used as a control device 200 for any network-based resource. Switches, hubs, routers, and other networking devices are candidates for a control device 200. The network-based resources they use could include any service that cannot physically reside on the networking device due to restrictions such as limited memory.
- Some embodiments of the invention include only the control device and the programs on the control device; other embodiments include some and/or all of the programs in computer readable media, or in electromagnetic waveforms.

V. Other Applications

- Integrated desktop/Palm email access. E.g., use the Palm to access the desktop's Outlook program to view/print an attachment.

- Calendar access and update: group calendaring and contacts information access. Integrate the Palm calendar/contacts with the desktop or enterprise calendar/contact systems. For example, the Palm may have only a subset of the enterprise's phone list, the Jini interface may support an expanded search (e.g. via an LDAP server) to search enterprise wide contacts.

- Substitute user interfaces: provide user interfaces to devices that do not have any, or good, user interfaces, e.g., hearing aid adjustment (note potential prior art that Elaine had mentioned), control system devices. The Palm could control network device configurations (e.g., switch/router), e.g., provide a telnet client in the Jini repository to allow the Palm to change a network device's configuration. Also, home network remote control is possible. This provides ease of use for controlling home network devices, gateways, home appliances, TVs etc.

- Desktop access/control: PCAnywhere like access that is controlled from the Palm. This could allow basic editing of documents, which maintains desktop application qualities of the documents. For example, use the Palm to fill out a Word document

form (Word is running on a desktop). This maintains the Word nature of the completed form. Thus, the user does not have to translate between document types.

- Typical network services: printing, faxing.
- Web access control: control web surfing.
- Group settings/management: e.g., controlling video conferencing/photo capture, controlling a shared electronic whiteboard, conference room reservation.
- Language and/or speech translation.

W. Conclusion

The foregoing description of various embodiments of the invention has been presented for purposes of illustration and description. It is not intended to limit the invention to the precise forms disclosed. Many modifications and equivalent arrangements will be apparent.

CLAIMS

What is claimed is:

1. A method of controlling a service on a network using a palm sized computer, the palm sized computer being coupled in communications with the network, the method comprising:

accessing a description of the service from a directory of services, the description of the service including at least a reference to program code for controlling the service;

downloading the program code to the palm sized computer;

the palm sized computer executing at least a portion of the program code; and

sending control commands to the service from the palm sized computer in response to the executing, wherein the service controls an application that cannot be executed on the palm sized computer.

2. The method of claim 1 further comprising registering the service in the directory of services by storing the description of the service in the directory of services.

3. The method of claim 1 wherein the directory of services includes a Jini Lookup directory, wherein the accessing the description includes executing a Jini discovery protocol to locate the Jini Lookup directory and executing a Jini Lookup protocol to retrieve the description of the service.

4. The method of claim 1 wherein the description of the service includes an object reference corresponding to an object representing the service and a set of service attributes including the name of the service and the physical location of the service.

5. The method of claim 1 wherein the program code includes Java code and wherein the palm sized computer is executing a Java Virtual Machine to execute at least a portion of the program code.

6. The method of claim 1 wherein the program code includes code to implement a graphical user interface on the palm sized computer.

7. The method of claim 1 wherein the application includes a desktop program.

8. A method of controlling a program on a network device from a computer, the computer is not capable of executing the program by itself, the network device and computer being coupled in communications via a network, the method comprising:

accessing a directory of services, a service in the directory of services corresponding to the program, the description of the service including at least a reference to program code for controlling the service;

loading the program code;

issuing control commands to the network device using the program code, the control commands causing the network device to control the program.

9. The method of claim 8 wherein loading the program code includes loading the program code onto the computer and the issuing the control commands includes the computer issuing the control commands.

10. The method of claim 8 wherein a proxy device is coupled to the network and wherein accessing the directory of services includes the computer accessing the proxy device, and the proxy device accessing the directory of services, and wherein the loading the program code includes loading the program code onto the proxy device, and wherein the issuing the control commands includes the computer issuing a set of first set of commands to the proxy device and the proxy device issuing the control commands.

11. The method of claim 10 wherein the program code includes a user interface program and wherein the proxy device receives the user interface program and generates a second user interface for the computer.

12. The method of claim 8 wherein loading the program code includes loading the program code onto the computer from the directory of services.

13. The method of claim 8 wherein the computer includes a Palm OS compatible computer, wherein the program code includes Java code and wherein the directory of services includes a Jini directory of services.

14. The method of claim 8 further comprising the network device registering the description of the service with the directory of services.

15. The method of claim 8 wherein the program includes a desktop program.

16. The method of claim 8 wherein the program includes a desktop program and wherein the services controlled include a CPU service for executing the program, a storage service for providing data to the CPU service and a display service for displaying information generated from the CPU service.

17. A system for controlling a network service comprising:

a network based computer service for controlling an application that cannot be executed by the control device;

a directory of services including a registry of services, each service in the registry of services corresponding to a service on the network;

a control device having a program for sending control commands to the network based computer service, the program being loaded onto the control device as a result of locating a reference to the network based computer service in the directory of services;

a network coupled in communications with the network based computer service, the directory of services and the control device.

18. The method of claim 17 wherein the control device includes a palm sized computer having an operating system, a network communications program, a protocol

program for communicating with the directory of services and wherein the program includes a graphical user interface.

19. The method of claim 17 wherein the network based computer service includes a computer having an operating system, a network communications program, a protocol program for communicating with the directory of services and the application.

20. A method for controlling a service in a network using a computer, the method comprising:

accessing a description of a service, the description of the service including at least a reference to program code for controlling the service;

downloading the program code;

executing at least a portion of the program code; and

sending control commands to the service in response to the executing, wherein the service includes a CPU service.

21. A system comprising:

means for accessing a description of a service, the description of the service including at least a reference to program code for controlling a service;

means for downloading the program code;

means for executing at least a portion of the program code; and

means for sending control commands to the service in response to the means for executing, wherein the service controls an application that cannot be executed on the means for executing.

22. A data processing tool for controlling an application accessible via a network, comprising:

a console application including a user interface program, information about services, including network addresses, in a group of services accessible via the network, and a communication driver executing a protocol for communication of the console application with at least one of the services in the group;

an input/output device supporting the user interface program; and
a communication port by which access to the network is available.

23. The data processing tool of claim 22, wherein the protocol includes an exchange in which the console application notifies a particular service in the group of services which will act as an application host, of a set of services to be invoked.

24. The data processing tool of claim 22, wherein the protocol includes an exchange by which the console application learns the network addresses of services in the group.

25. The data processing tool of claim 22, wherein the protocol includes an exchange in which a particular service in the group of services sends the console application a set of user interface constructs for incorporation in the user interface program.

26. The data processing tool of claim 22, wherein a particular service in the group of services comprises a slide presentation program.

27. The data processing tool of claim 22, wherein the particular service in the group of services comprises an email client program.

28. The data processing tool of claim 23, wherein the particular service in the group of services comprises a calendar program.

29. The data processing tool of claim 22, wherein the particular service in the group of services comprises a user interface program for an networked appliance.

30. The data processing tool of claim 22, wherein the particular service in the group of services comprises a print service.

31. The data processing tool of claim 22, wherein the particular service in the group of services comprises a fax service.

32. The data processing tool of claim 22, wherein the particular service in the group of services comprises an internet browser service.

33. The data processing tool of claim 22, wherein the particular service in the group of services comprises a language and/or speech translation service.

34. The data processing tool of claim 22, wherein the particular service in the group of services comprises a conference room reservation function.

35. The data processing tool of claim 22, wherein the port comprises a wireless transmitter and receiver.

36. The data processing tool of claim 22, wherein the port comprises an infrared transmitter and receiver.

37. The data processing tool of claim 22, wherein the input/output device comprises a touch screen.

38. The data processing tool of claim 22, wherein the input/output device comprises a touch screen smaller than 4 inches by 6 inches in display area.

39. A method for controlling an application executable on a particular processor coupled to a network using a portable computing platform, comprising:

establishing a communication link via the network between the portable computing platform and the particular processor;

transferring a control program to the portable computing platform via the network, the control program including user interface constructs for generating commands for control of the application;

transmitting commands input using the control program to the particular processor via the communication link;

transferring the commands input using the control program to the application.

40. The method of claim 39, wherein the application comprises a slide presentation application, and the commands input using the control program include commands for opening a presentation for display on a display coupled to the network, under control of the particular processor, and navigating slides within the presentation.

41. The method of claim 39, wherein the application comprises a slide presentation application, and the commands input using the control program include commands for editing slides within the presentation.

42. The method of claim 39, wherein the communication link comprises a wireless link.

43. The method of claim 39, wherein the communication link comprises an infrared link.

44. The method of claim 39, wherein portable computing platform includes a touch screen, and the interface constructs include graphical user interface elements accepting inputs via the touch screen.

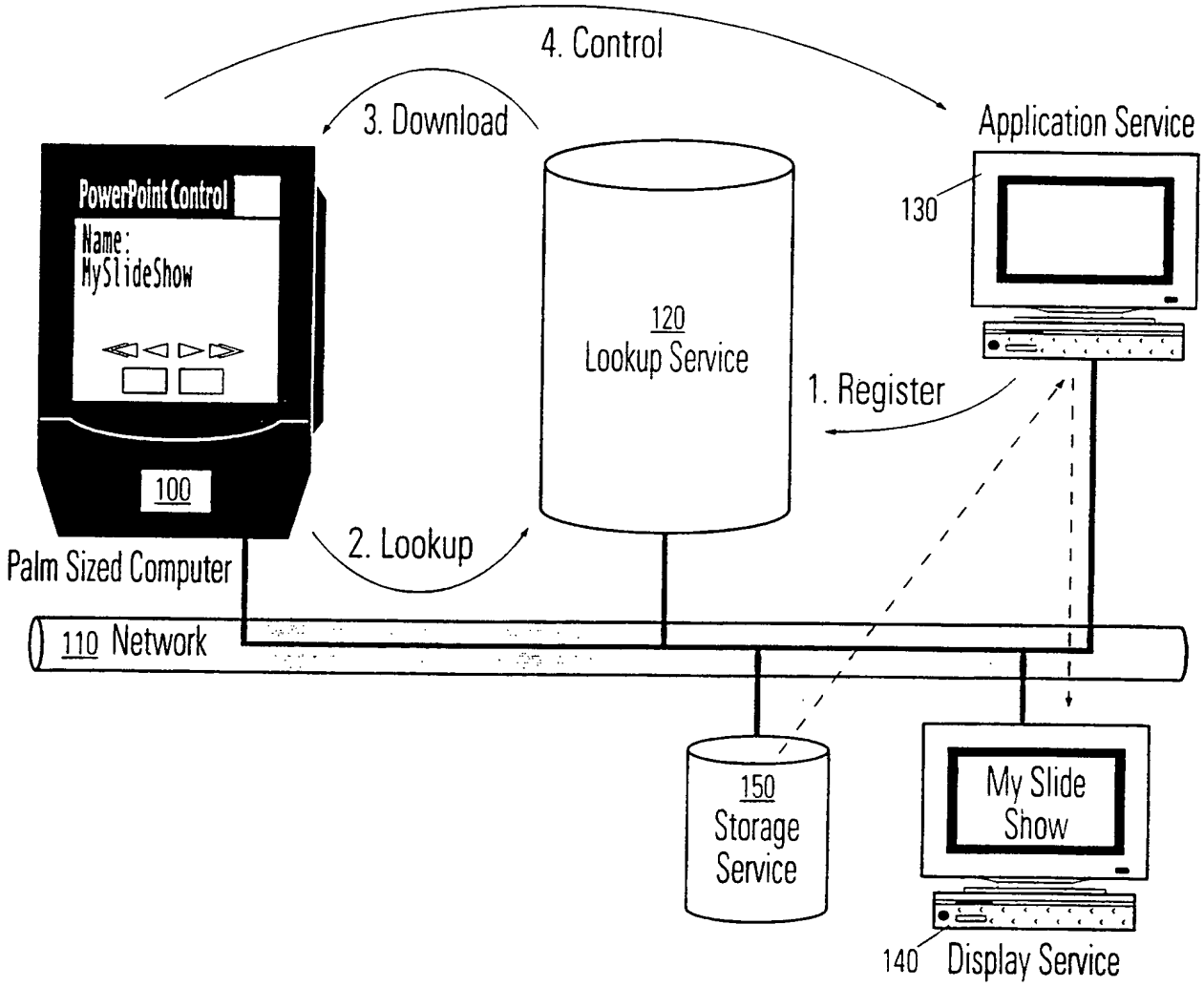

4. Control
3. Download
Application Service
PowerPoint Control
Name:
MySlideShow
130
120
Lookup Service
1. Register
100
2. Lookup
Palm Sized Computer
110 Network
150
Storage
Service
My Slide
Show
140 Display Service

FIG. 1

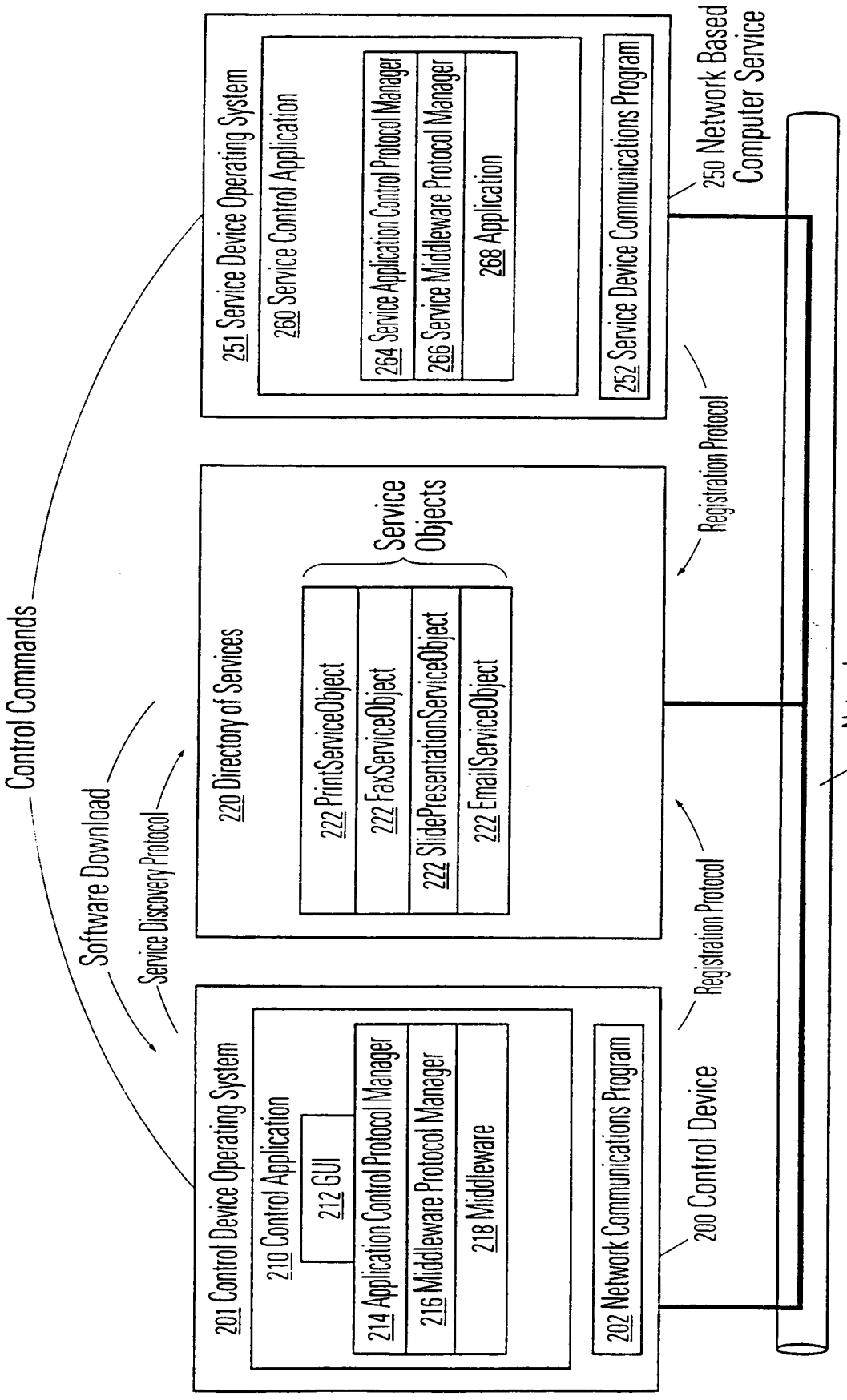

Control Commands
Software Download
Service Discovery Protocol
201 Control Device Operating System
210 Control Application
212 GUI
214 Application Control Protocol Manager
216 Middleware Protocol Manager
218 Middleware
202 Network Communications Program
200 Control Device
220 Directory of Services
Service Objects
222 PrintServiceObject
222 FaxServiceObject
222 SlidePresentationServiceObject
222 EmailServiceObject
Registration Protocol
Registration Protocol
251 Service Device Operating System
260 Service Control Application
264 Service Application Control Protocol Manager
266 Service Middleware Protocol Manager
268 Application
252 Service Device Communications Program
250 Network Based Computer Service
110 Network

FIG. 2

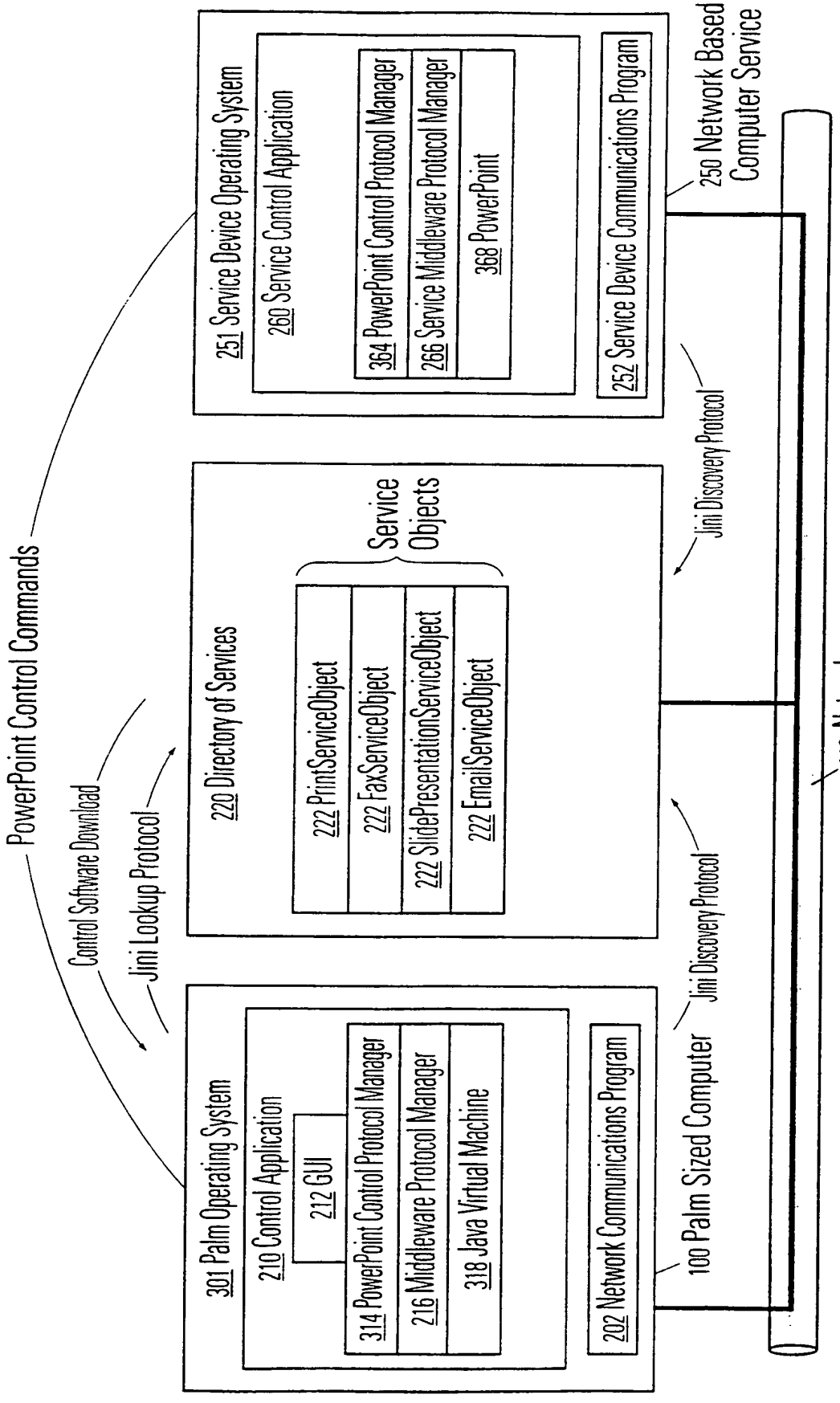

PowerPoint Control Commands
Control Software Download
Jini Lookup Protocol
301 Palm Operating System
210 Control Application
212 GUI
314 PowerPoint Control Protocol Manager
216 Middleware Protocol Manager
318 Java Virtual Machine
202 Network Communications Program
100 Palm Sized Computer
Jini Discovery Protocol
220 Directory of Services
Service Objects
222 PrintServiceObject
222 FaxServiceObject
222 SlidePresentationServiceObject
222 EmailServiceObject
Jini Discovery Protocol
251 Service Device Operating System
260 Service Control Application
364 PowerPoint Control Protocol Manager
266 Service Middleware Protocol Manager
368 PowerPoint
252 Service Device Communications Program
250 Network Based Computer Service
110 Network

FIG. 3

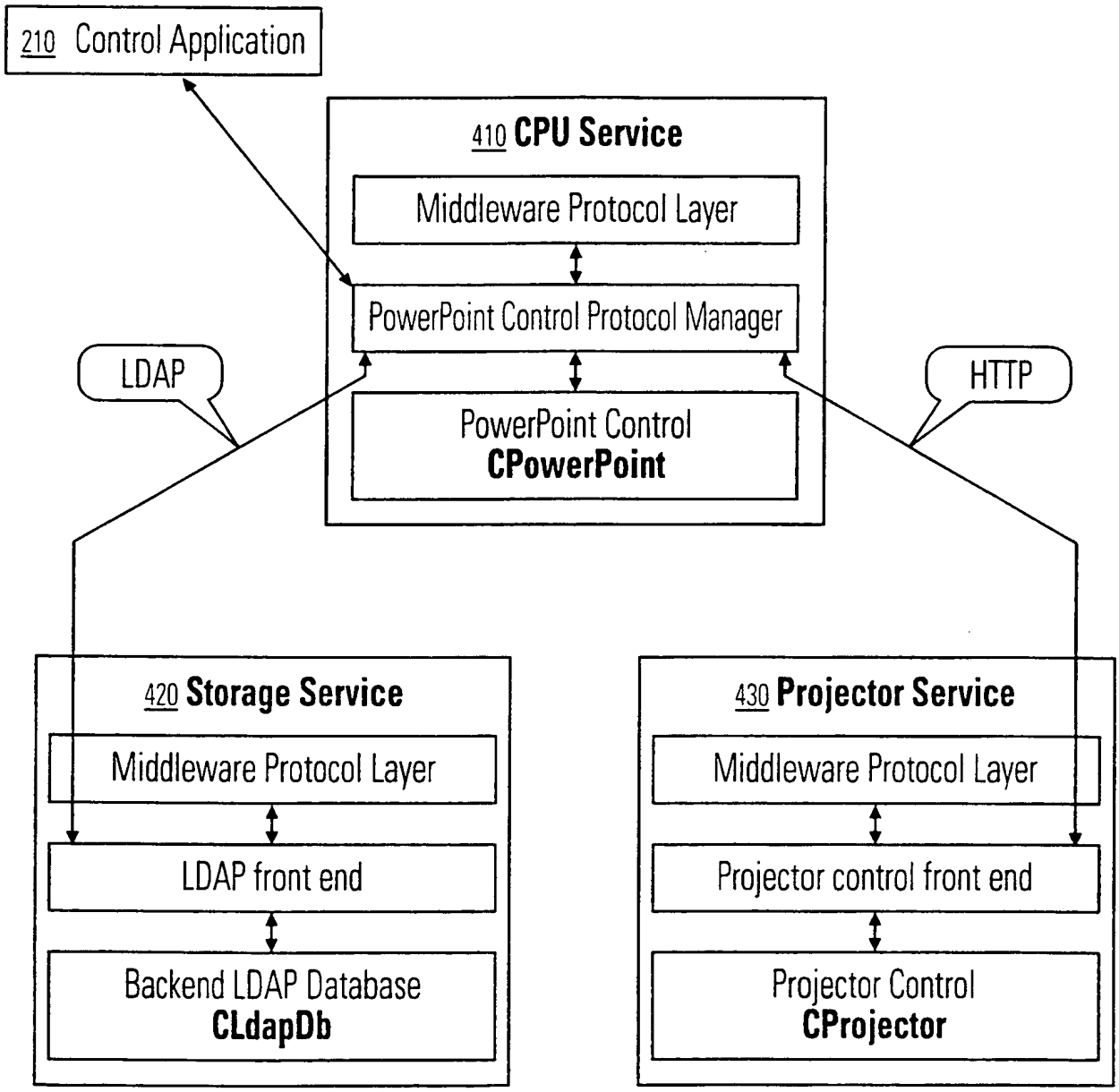

210 Control Application
410 CPU Service
Middleware Protocol Layer
PowerPoint Control Protocol Manager
PowerPoint Control
CPowerPoint
LDAP
HTTP
420 Storage Service
Middleware Protocol Layer
LDAP front end
Backend LDAP Database
CLdapDb
430 Projector Service
Middleware Protocol Layer
Projector control front end
Projector Control
CProjector

FIG. 4

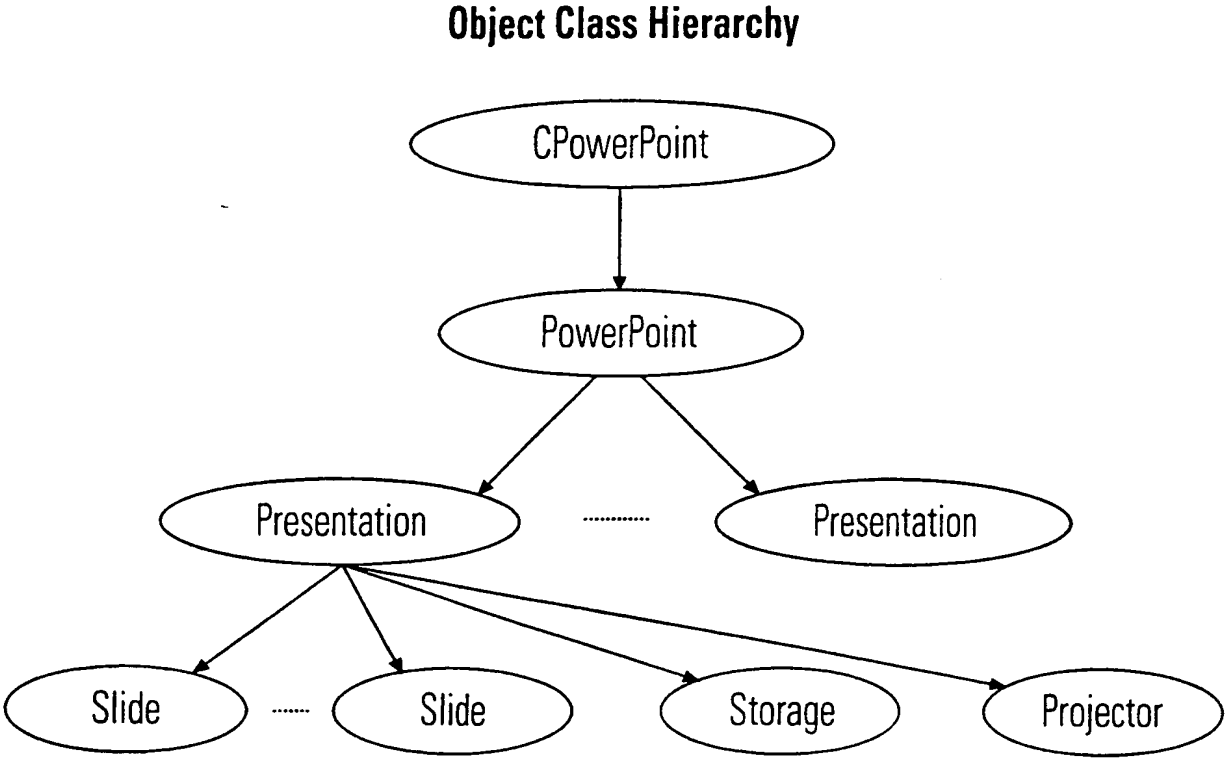

Object Class Hierarchy
CPowerPoint
PowerPoint
Presentation
Presentation
Slide
Slide
Storage
Projector

FIG. 5

Object Class Hierarchy for Network Services

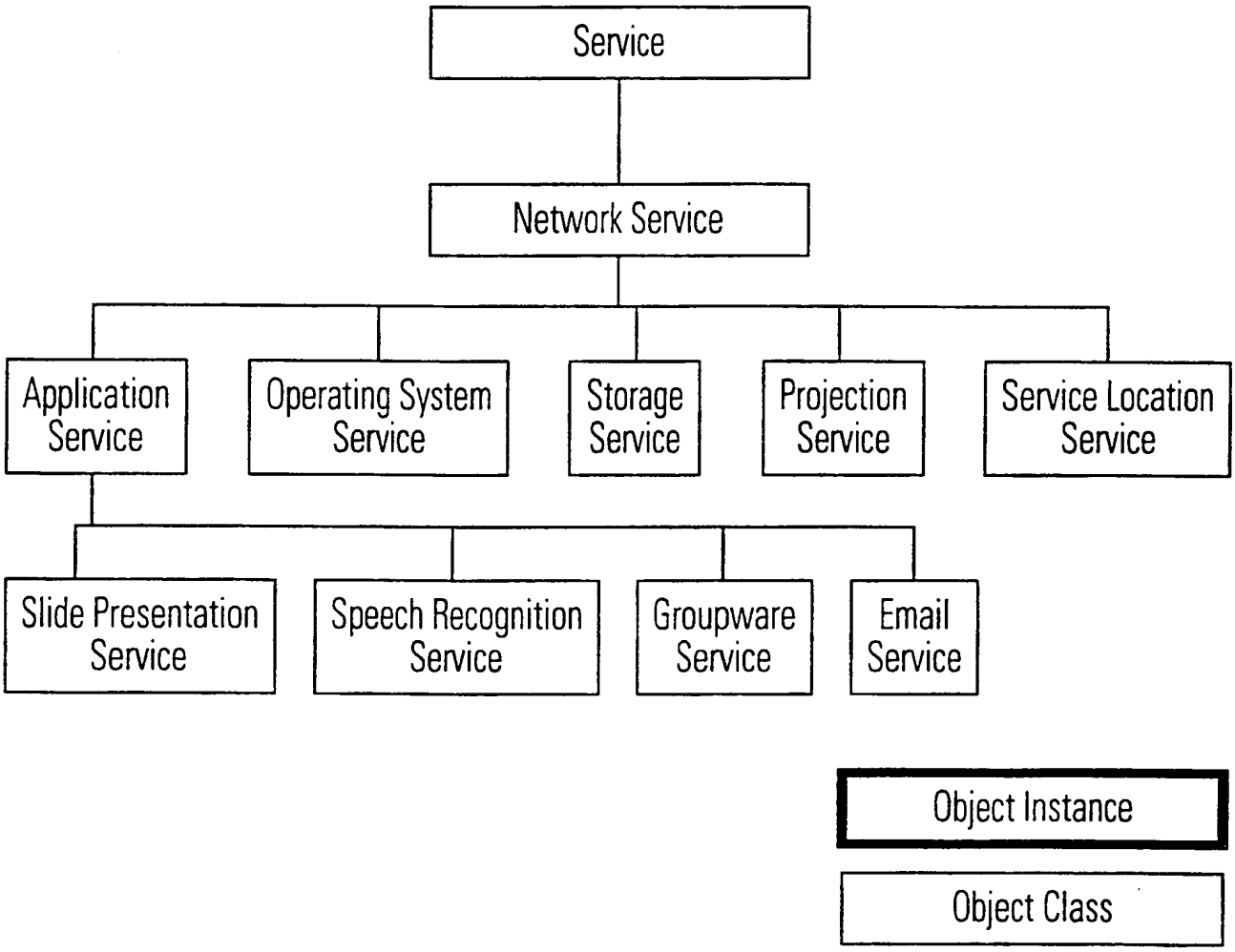


FIG. 6

Object Class Hierarchy for Application Services

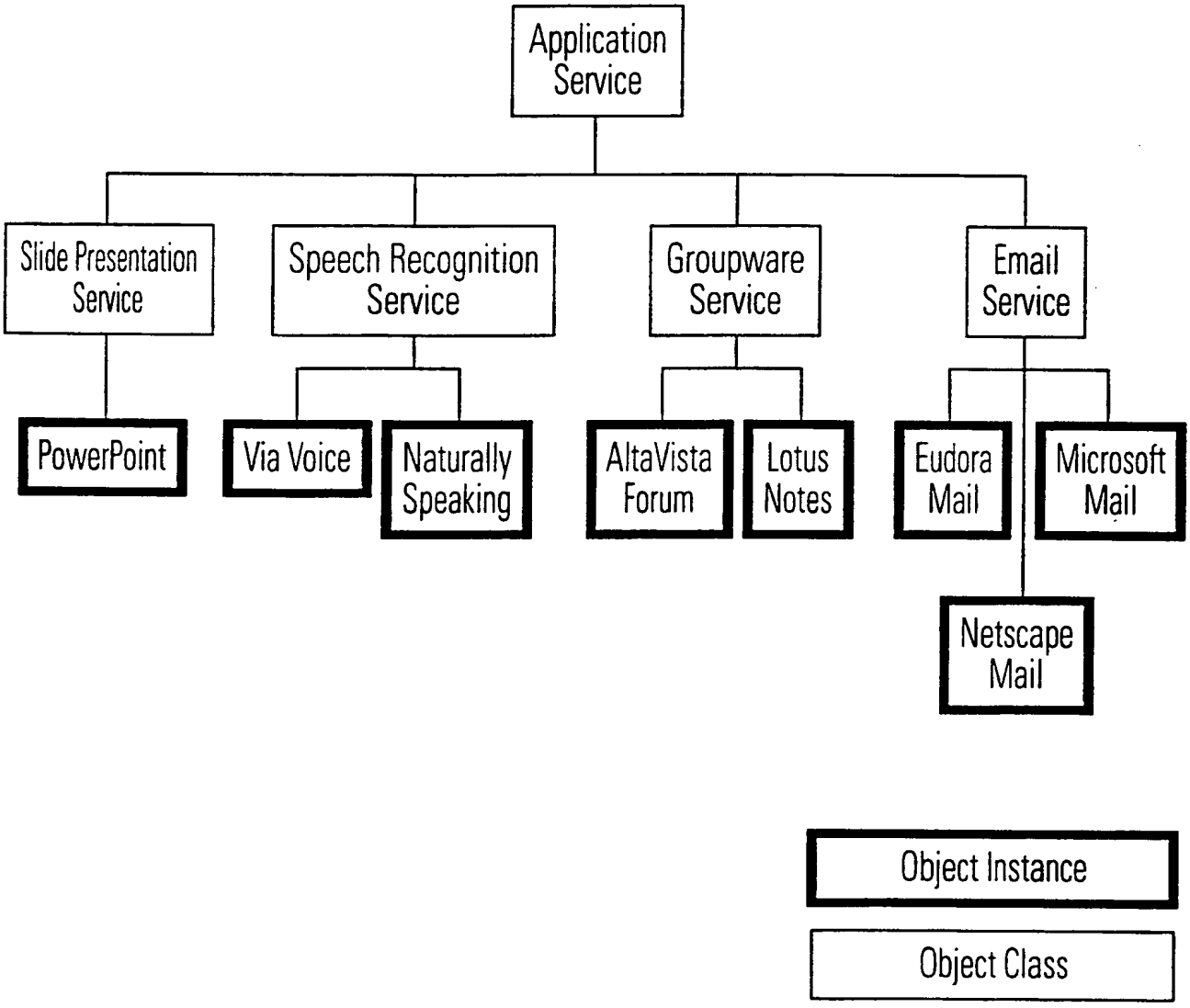


FIG. 7

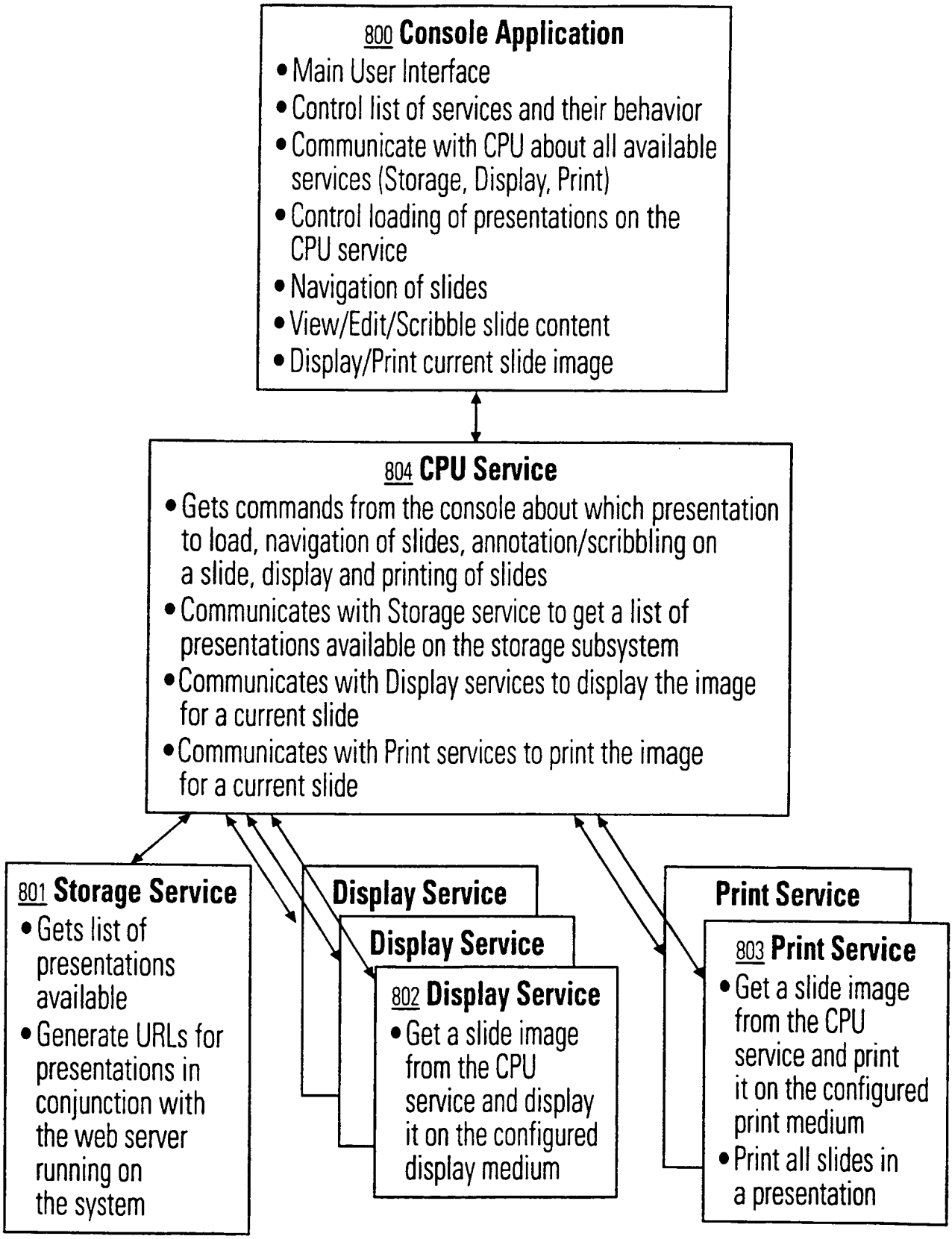

800 Console Application
• Main User Interface
• Control list of services and their behavior
• Communicate with CPU about all available services (Storage, Display, Print)
• Control loading of presentations on the CPU service
• Navigation of slides
• View/Edit/Scribble slide content
• Display/Print current slide image
804 CPU Service
• Gets commands from the console about which presentation to load, navigation of slides, annotation/scribbling on a slide, display and printing of slides
• Communicates with Storage service to get a list of presentations available on the storage subsystem
• Communicates with Display services to display the image for a current slide
• Communicates with Print services to print the image for a current slide
801 Storage Service
• Gets list of presentations available
• Generate URLs for presentations in conjunction with the web server running on the system
Display Service
Display Service
802 Display Service
• Get a slide image from the CPU service and display it on the configured display medium
Print Service
803 Print Service
• Get a slide image from the CPU service and print it on the configured print medium
• Print all slides in a presentation

FIG. 8

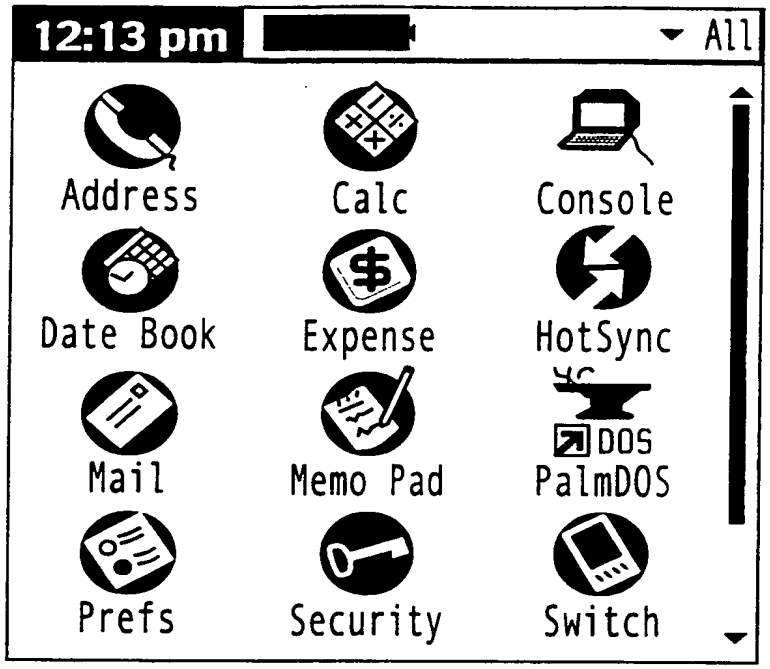

12:13 pm
All
Address
Calc
Console
Date Book
Expense
HotSync
Mail
Memo Pad
DOS
PalmDOS
Prefs
Security
Switch

FIG. 9

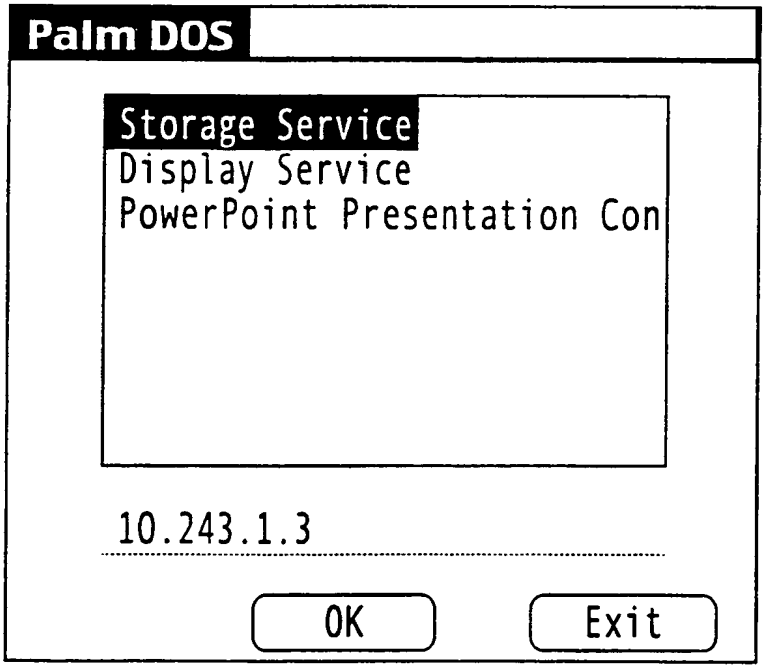

Palm DOS
Storage Service
Display Service
PowerPoint Presentation Con
10.243.1.3
OK
Exit

FIG. 10

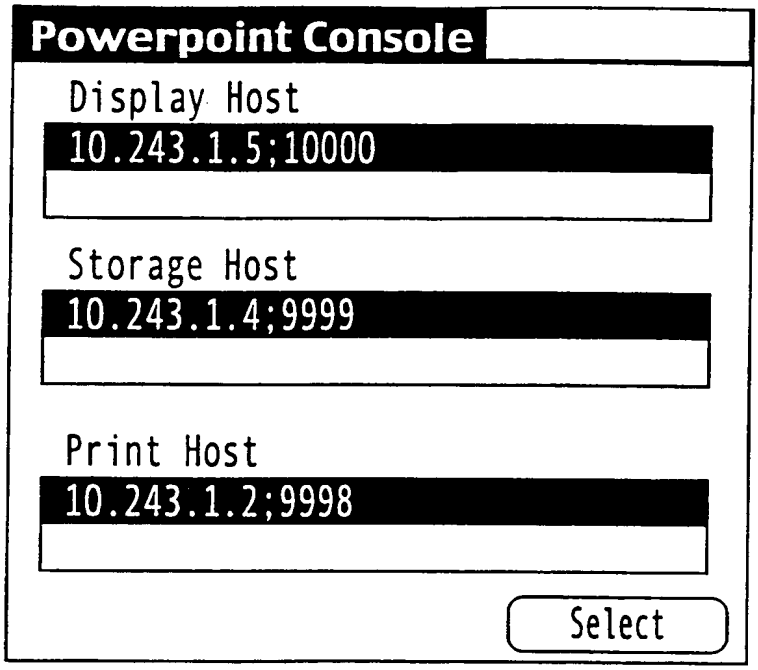

Powerpoint Console
Display Host
10.243.1.5;10000
Storage Host
10.243.1.4;9999
Print Host
10.243.1.2;9998
Select

FIG. 11

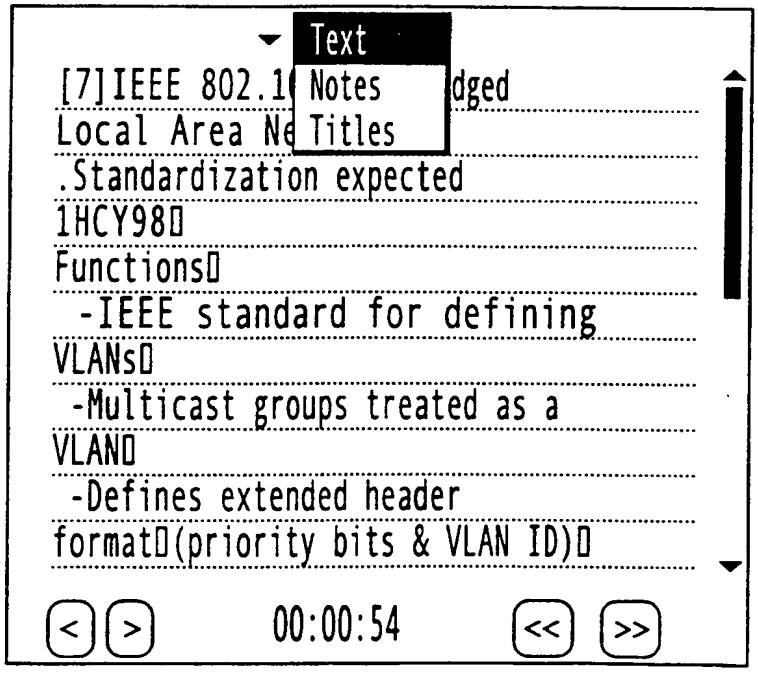

Text
Notes
Titles
[7]IEEE 802.1(dged
Local Area Ne
.Standardization expected
1HCY98
Functions
-IEEE standard for defining
VLANs
-Multicast groups treated as a
VLAN
-Defines extended header
format (priority bits & VLAN ID)
00:00:54

FIG. 12

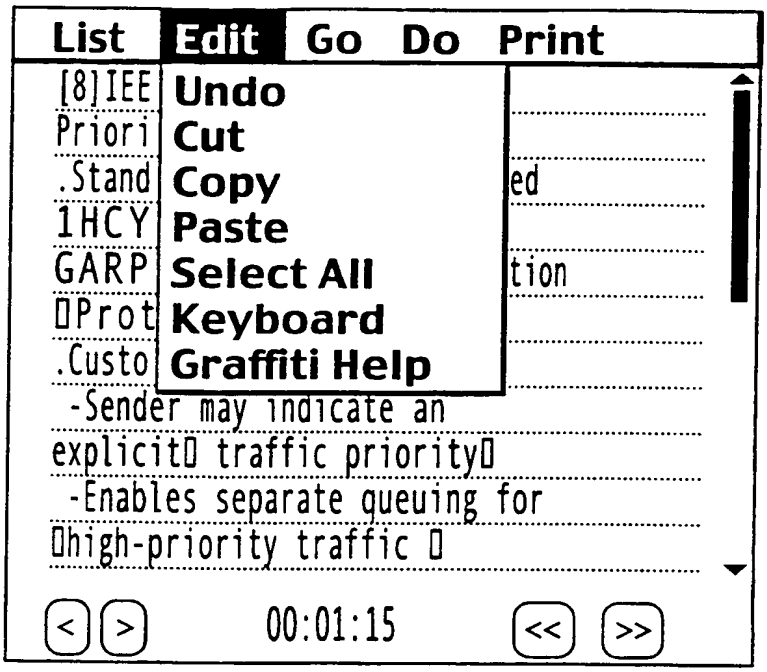

List
Edit
Go
Do
Print
Undo
Cut
Copy
Paste
Select All
Keyboard
Graffiti Help
[8]IEE
Priori
.Stand
1HCY
GARP
Prot
.Custo
ed
tion
-Sender may indicate an
explicit traffic priority
-Enables separate queuing for
high-priority traffic
00:01:15

FIG. 13

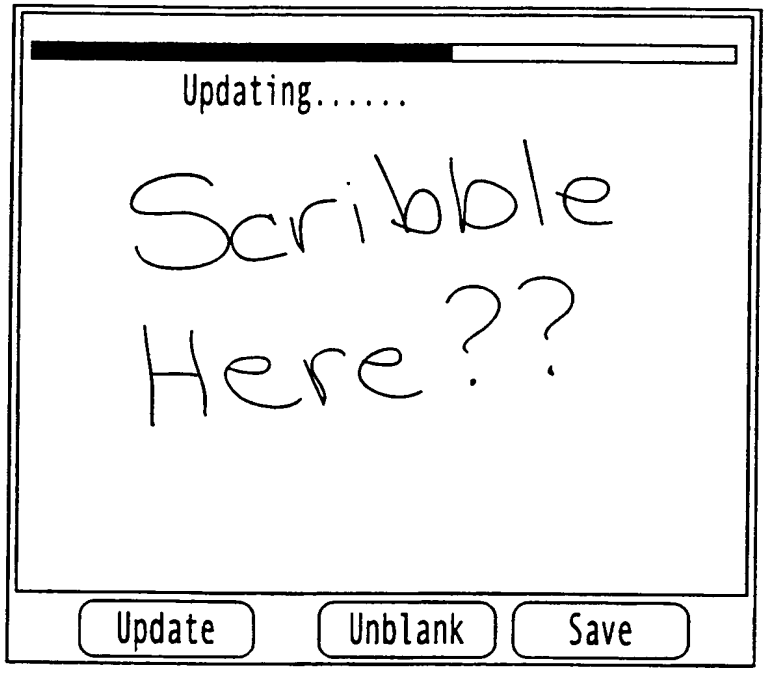

Updating.....
Scribble
Here??
Update
Unblank
Save

FIG. 14

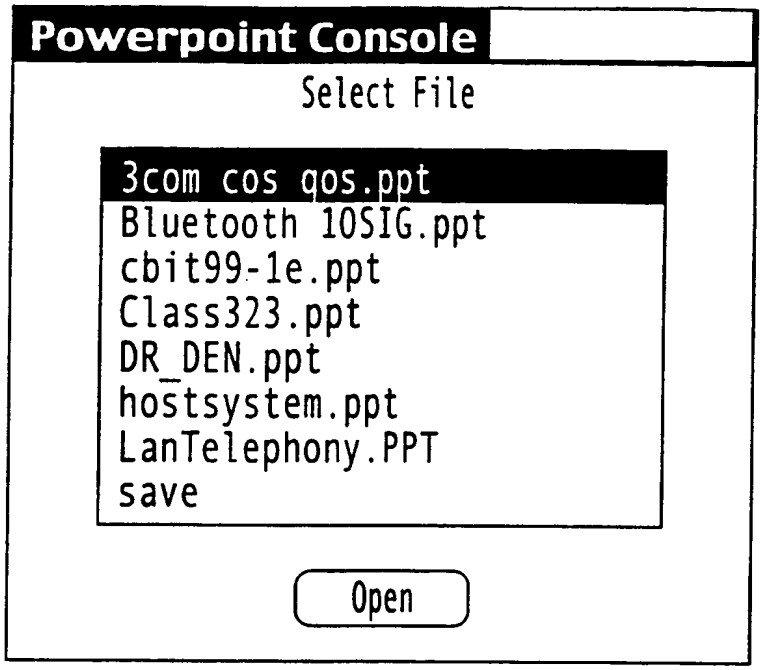

Powerpoint Console
Select File
3com cos qos.ppt
Bluetooth 10SIG.ppt
cbit99-1e.ppt
Class323.ppt
DR_DEN.ppt
hostsystem.ppt
LanTelephony.PPT
save
Open

FIG. 15

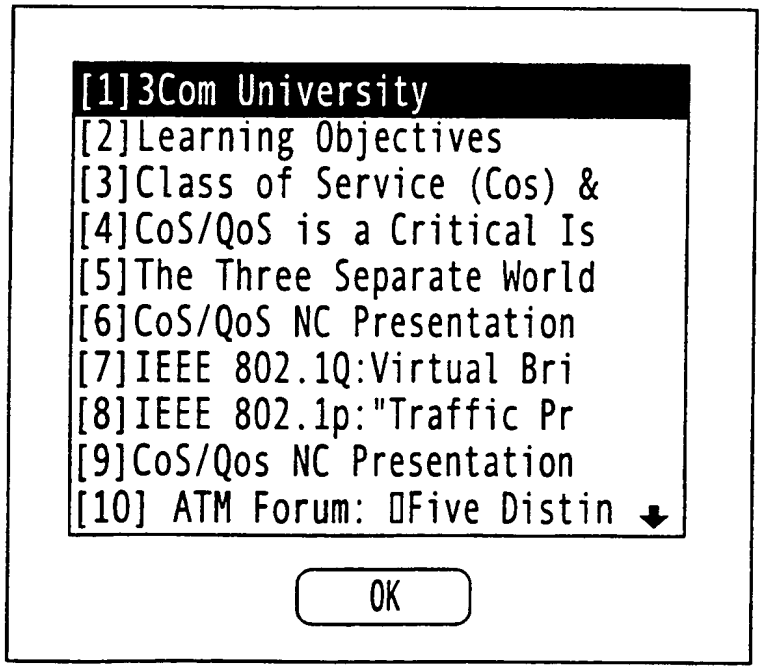

[1]3Com University
[2]Learning Objectives
[3]Class of Service (Cos) &
[4]CoS/QoS is a Critical Is
[5]The Three Separate World
[6]CoS/QoS NC Presentation
[7]IEEE 802.1Q:Virtual Bri
[8]IEEE 802.1p:"Traffic Pr
[9]CoS/Qos NC Presentation
[10] ATM Forum: □Five Distin
OK

FIG. 16